Uppbygnad av virtuell panoramapresentation av Arcadas utrymmen genom att använda Papervision3D

Mikael Lindberg

Mikael Lindberg

Mikael Lindberg

EXAMENSARBETE	
Arcada	
Utbildningsprogram:	Informations- och medieteknik
Identifikationsnummer:	3235
Författare:	Mikael Lindberg
Arbetets namn:	Uppbyggnad av virtuell panoramapresentation av Arcadas utrymmen genom att använda Papervision3D
Handledare (Arcada):	Johnny Biström
Uppdragsgivare:	Arcada

Sammandrag:

Detta examensarbete består av två huvuddelar. Den första delen är en teoretisk analys av hur 3D motorn Papervision3D är uppbyggd. Den andra delen av arbetet är en praktisk genomgång av hur panoramapresentationen är förverkligad.

Den teoretiska analysen undersöker hur Papervision3D grundades och av vilka huvudklasser 3D motorn är uppbyggd. Papervision3D:s uppbyggnad och egenskaper pejlas för att få en förståelse för hur man skall kunna utnyttja motorn för praktiska webbapplikationer. Även om utvecklandet av Papervision3D har inofficiellt slutat så kan man fortfarande använda motorn. Användning medför trots allt vissa risker.

Den praktiska delen består av kravspecifikation samt de essentiella arbetsskeden för att fotografera och bygga panoramapresentationen. Programmeringen är upplyst av kodexempel för att poängtera betydelsen. I den praktiska delen behandlas även hur eye tracking användes i examensarbetet.

Nyckelord:	Papervision3D, 3D, Adobe Flash, Eye Tracking, Panorama, Presentation
Sidantal:	58
Språk:	Svenska
Datum för godkännande:	

DEGREE THESIS	
Arcada	
Degree Programme:	Information- and mediatechnology
Identification number:	3235
Author:	Mikael Lindberg
Title:	Construction of a virtual panoramic presentation of Arcada facilities by using Papervision3D
Supervisor (Arcada):	Johnny Biström
Commissioned by:	Arcada

Abstract:

This thesis consists of two main parts. The first part is a theoretical analysis of the 3D engine Papervision3D and how it is constructed. The second part of the thesis is a practical review of how the panoramic presentation is realized.

The theoretical analysis examines how Papervision3D was founded and of which main classes the 3D engine is built. Papervision3D's structure and effects are explored to gain an understanding of how to utilize the engine as practical web applications. Although the development of Papervision3D has unofficially ended, it is still fully possible to use the engine. Using can however envolve some risks.

The practical part consists of requirement specification, and the essential phases of constructing a panoramic presentation. Programming is highlighted with code examples to point out the significance. The practical part also includes how eye tracking was used in this thesis.

Keywords:	Papervision3D, 3D, Adobe Flash, Eye Tracking, Panorama, Presentation
Number of pages:	58
Language:	Swedish
Date of acceptance:	

OPINNÄYTE	
Arcada	
Koulutusohjelma:	Informaatio- ja mediatekniikka
Tunnistenumero:	3235
Tekijä:	Mikael Lindberg
Työn nimi:	Virtuaalisen panoraamaesityksen rakentaminen Arcadan tiloista käyttäen Papervision3D:tä
Työn ohjaaja (Arcada):	Johnny Biström
Toimeksiantaja:	Arcada

Tiivistelmä:

Tämä opinnäyte koostuu kahdesta osasta. Ensimmäinen osa on teoreettinen analyysi Papervision3D-moottorista ja sen rakenteesta. Toinen osa opinnäytteestä on käytännön katsaus miten panoraamaesitys on toteutettu.

Teoreettisessä analyysissä tutkitaan Papervision3D:n historiaa ja mistä pääluokista moottori koostuu. Papervision3D:n ominaisuudet ja rakenne on koottu, jotta siitä saisi ymmärryksen siitä, miten käyttää moottoria käytännössä eri web-sovelluksissa. Vaikka Papervision3D:n kehitys on epävirallisesti lopetettu, se ei estä sen käyttämista nykypäivässä. Käyttö tuo kuitenkin omat riskinsä mukaan.

Käytännön osuudessa käydään läpi panoraamaesityksen vaatimukset sekä olennaiset työvaiheet. Ohjelmointi on valaistu esimerkkikoodilla korsostaakseen sen merkitystä. Käytännön osuudessa käsitellään myös eye trackerin käyttö opnnäytteessä.

Avainsanat:	Papervision3D, 3D, Adobe Flash, Eye Tracking, Panoraama, Esitys
Sivumäärä:	58
Kieli:	Ruotsi
Hyväksymispäivämäärä:	

INNEHÅLL

1 Inledning 11

1.1 Introduktion 11
1.2 Syfte och målsättning 11
1.3 Avgränsning 11
1.4 Begrepp 12

2 Papervision3D 13

2.1 Rättigheter 13
2.2 Grundare 13
2.3 Historia 14
2.4 Konkurrenter 15
2.4.1 Away3D 16
2.4.2 Sandy 3D 16
2.4.3 Sophie 3D 17
2.4.4 Alternativa3D 17
2.5 Framtid 18

3 Papervision3D egenskaper 19

3.1 Initiera Papervision 19
3.1.1 Versioner 19
3.1.2 Installering och användning 20
3.2 Papervision3D huvudklasser 20
3.2.1 Scene3D 21
3.2.2 Viewport3D 21
3.2.3 Camera3D 22
3.2.4 Rendering 24
3.2.5 DisplayObject3D 25
3.3 BasicView 25
3.4 Primitiver 26
3.4.1 Plan 27
3.4.2 Kub 27
3.4.3 Sfär 28
3.4.4 Cylinder 29
3.4.5 Kon 29
3.5 Material 30
3.5.1 Färgmaterial 30
3.5.2 Bitmapmaterial 31

3.5.3 Filmmaterial ... *32*
3.5.4 Skuggor och skuggning ... *32*
3.6 Interaktivitet ... 33

4 Kravspecifikation ... 34

4.1 Kravinsamling ... 34
4.1.1 Intervjuer ... *34*
4.2 Eye Tracker ... 36
4.2.1 Eye Tracking ... *36*
4.2.2 Resultat ... *37*
4.2.3 Diskussion ... *39*

5 Praktiska utförandet ... 40

5.1 Programmering ... 40
5.1.1 Arbetsverktyg ... *40*
5.1.2 Skapandet av presentationen ... *41*
5.1.3 Interaktivitet ... *42*
5.1.4 Grafik ... *45*
5.2 Fotografering ... 47
5.2.1 Utrustning och arbetsmetoder ... *47*
5.2.2 Fotografering ... *49*
5.2.3 Ihopmontering ... *49*
5.2.4 Redigering ... *50*
5.3 Uppbyggnad ... 52

6 Diskussion ... 53

6.1 Papervision3D ... 53
6.1.1 Fördelar ... *54*
6.1.2 Nackdelar ... *54*
6.2 Presentationen ... 55
6.3 Slutsats ... 56

Källor ... 57

Bilagor ... 59

Bilaga 1. Intervjufrågor

Bilaga 2. Eye Tracking forskningsfrågor

Bilaga 3. Fullständig källkod över panoramapresentationen

Bilaga 4. Resultat över Eye Tracking testet (DVD)

Figurer

Figur 1. Visualisering över de essentiella delarna i Papervision3D enligt Winder & Tondeur (2009 s. 38)..... 21
Figur 2. Visualisering över kamera och dess begränsningar (Lively 2010 s.38)..... 23
Figur 3. Exempel på olika plan primitiver. 27
Figur 4. Exempel på olika kub primitiver. 27
Figur 5. Exempel på olika sfär primitiver. 28
Figur 6. Exempel på olika cylinder primitiver. 29
Figur 7. Exempel på olika kon primitiver. 29
Figur 8. Exempel på olika färgmaterial. 30
Figur 9. Exempel på olika Wireframe material 31
Figur 10. Exempel på användning av ShadowCaster-klassen (Zupko 2008b). 33
Figur 11. Exempel på Flat-, Cell-, Gouraud-, Phong-, och EnvMapShader (Kallonen 2010). 33
Figur 12. Resultatredovisning över fråga 8. 35
Figur 13. Tobii T120 Eye Tracker (Tobii T120 2010). 36
Figur 14. Värmekarta över de populäraste områden i presentationen. 38
Figur 15. Antal preciseringar. 39
Figur 16. Tid på besök. 39
Figur 17. Exempel på hur man får information i ett litet fönster i programmet Adobe Flash Builder. 41
Figur 18. Meny. 45
Figur 19. Exempel på 6 exponeringar. 48
Figur 20. Exempel på programmet PTgui. 50
Figur 21. Jämförelse mellan vanlig- och HDR-fotografi. 51
Figur 22. Exempel på nadir före och efter redigering. 52
Figur 23. Panoramapresentationens framsida. 53
Figur 24. Slutvärdering över panoramapresentation. 55

Förkortningar

API	Application Programming Interface
HTML	Hypertext Markup Language
MIT	Massachusetts Institute of Technology
HDR	High Dynamic Range
MXML	Macromedia Extensible Markup Language
TFT	Thin-Film Transistor

FÖRORD

Skrivandet av detta examensarbete har varit en stor utmaning.

Jag vill tacka företaget Testure för deras understöd under detta projekt. Jag vill också tacka min fästmö Camilla Byman samt min syster Minna Lindberg som varit till stor hjälp då jag skrev denna rapport.

1 INLEDNING

1.1 Introduktion

Detta examensarbete är gjort som ett projekt för Arcada. Projektet gick ut på att bygga en presentation för de viktigaste utrymmen Arcada hyr ut. Materialet består av panoramafotografier. Presentationen är förverkligad i Adobe Flash Builder genom att använda Papervision3D motor.

1.2 Syfte och målsättning

Syftet med presentationen är att väcka intresset och höja försäljningen av utrymmen i Arcada. Dessutom förväntas att eventuella nya studeranden kan ha nytta av att bekanta sig med Arcadas utrymmen.

Målsättningen med examensarbetet var att identifiera om Papervision3D motor passar för uppbyggnad av panoramapresentation och hur användare närmar sig med panoramapresentationer.

Examensarbetet är uppdelat i två huvuddelar; en teoretisk och en praktisk. Den teoretiska delen består av beskrivning av bakgrund samt de viktigaste egenskaperna i Papervision3D, medan den praktiska delen består av hur själva presentationen är uppbyggd och följer de essentiella arbetsdelarna.

1.3 Avgränsning

Detta examensarbete kommer inte att omfatta hurdan kamerautrustning som är optimal för panoramafotograferandet. Istället kommer arbetet att fokusera på hur man effektivt kan utnyttja panoramabilder.

1.4 Begrepp

Collada

Collada är en standard för hantering av 3D miljöer.

ActionScript

Adobe Flash programmerings språk. Senaste versionen ligger på 3.0 (ActionScript3).

Rendera

Räkning och framställning av tredimensionella bilder till en tvådimensionell skärm. Vid komplicerade räkningar tar renderingen en stor del av effekten.

Lynda.com

On-line bibliotek för mjukvara utbildning och exempelvideon.

Primitiver

Gemensamt ord för de olika figurer i Papervision3D. En primitiv består av matematiska formler för olika figurer, till dem tillhör bl.a. kub, cylinder och sfär.

TweenLite

TweenLite är ett externt animeringsverktyg som sköter animering för olika objekt mycket bekvämt. Animeringsverktyget är programmerat av GreenSock och är gratis att använda.

Zenit och **Nadir**

Zenit och Nadir är riktningar och varandras motsatser. Zenit är vinkelrät rakt upp och Nadir respektive ner.

Eye Tracker

En apparat som registrerar ögats fokusering på skärmen. Via en Eye Tracker kan man med stor noggrannhet bestämma var människoögat tittar. Man kan till och med definiera i vilken ordning man läser ord i meningar.

2 PAPERVISION3D

Papervision3D är ett klassbibliotek för Adobe Flash för att skapa och visa 3D figurer i en tvådimensionell skärm. Paervision3D består av över 300 klasser. Genom att använda dessa klasser kan man skapa realistiska figurer i en 3D miljö. Papervision3D klassbibliotek är även kallad för en 3D motor.

2.1 Rättigheter

Papervision3D är öppen källkod och publicerad under MIT-licensen. Detta betyder att man får använda och modifiera källkoden till privat och kommersiellt bruk. Det är gratis att använda Papervision3D. (MIT Licence 2011)

2.2 Grundare

Papervision3D utvecklas och modifieras konstant för att uppfylla de nyaste kraven. Författare och utvecklare består av två mindre grupper; kärnutvecklare och hjälputvecklare med respektive special områden. Förutom detta, får de stöd från Papervisions utvecklar e-post kedja samt utvecklar samfundet som består av hundratals personer världen över. (Allen et al. 2008 s. 292)

Papervisions kärnutvecklare består av fem utvecklare och har ansvaret över alla de centrala delarna av Papervision projektet: utgivningarna, teamarbetet samt allmänt över API:n (Application Programming Interface). Kärutvecklargruppen består av:

Carlos Ulloa:	Grundare och ledare av Papervision3D.
John Grden:	Komponent- och utgivningsansvarig.
Ralph Hauwert:	Ansvarar för motorns kärna samt skuggning.
Tim Knip:	Ansvarar för motorns kärna, skuggning samt Collada relaterade ämnen.
Andy Zupko:	Ansvarar för effekterna.
Ben Hokins:	Ansvarar för motorns kärna.

(Allen et al. 2008 s. 292, Ulloa 2009)

Hjälputvecklare har rättighet att modifiera källkoden. Dessa personer är specialiserade på eget respektive område som faller utanför de kärnrelaterade frågorna. Hälputvecklargruppen utgörs av:

Ricardo Cabello:	Testing samt uppbyggnad av demonstrationsprogram.
De'Angelo Richardson:	Testning samt extra funktioner.
Stephen Downs:	Flex komponenter samt effekter.
Seb Lee Delisle:	Partiklar samt effekter.
John Lindquist:	Dokumentering samt exempel program.

(Allen et al. 2008 s. 292)

2.3 Historia

Carlos Ulloa började utvecklandet av Papervision i november 2005. Inspiration fick han av Joost Korngold efter ett seminarium i Amsterdam. Redan i januari 2006 hade Ulloa utvecklat ett klassbibliotek med vilken han kunde positionera, rotera, skala och skew-justera MovieClip objekt. (Allen et al. 2008 s. 291 f.)

I augusti 2006 publicerade Ulloa en ny version av sitt bibliotek med namnet Papervision3D. Biblioteket var fullt optimerat för att använda Flash 8 drawing API. Detta gjorde det möjligt att använda triangel tesslation för att bättre lägga textur på polygoner och skapa verkliga 3D objekt i Flash. Tesslation är utfyllnad av ett plan med geometriska figurer utan överlappning eller mellanrum. Tesslation kan vara att t.ex. att mura en vägg. (Tessellation 2011) Under de följande tio månaderna utvecklades Papervision till ett 14 klassers bibliotek och version 1.0 publicerades. (Allen et al. 2008 s. 291 f.)

I detta skede försnabbades utvecklingen av Papervision3D avsevärt. En ny webbsida och blogg skapades. Dessutom lades Papervision till webbsidan OSFlash.org, som presenterar öppenkällkod. Då anslöt sig också den första utomstående personen, John Grden, med i projektet. Han konverterade biblioteket till ActionScript 3, vilket gav en stor förbättring i prestationsförmågan. Hundratals programmerare laddade ner projektet och gav sitt eget bidrag till det nya fenomenala projektet. (Allen et al. 2008 s. 291 f.)

I början av 2007 anslöt sig Ralph Hauwert till teamet. Hans tekniska exempelprogram och material gjorde stort intryck på samfundet och hämtade ny publicitet för Papervision. Samtidigt publicerade John Grden det första spelet gjort med Papervision. I spelet flyger spelaren ett X-Wing fartyg i en bana. Strax efter detta publicerades de första kommersiella webbplatserna byggda på Papervision. (Allen et al. 2008 s. 291 f., Kallonen 2010 s. 3)

Samma år använde Tim Kinp flera månader på att utveckla kopplingen med Collada formatet med Papervision3D. Då detta var gjort anslöt sig Knip officiellt med i Papervisions utvecklar team. Ett år efter första versionen av Papervision utkom, lanserades alfa versionen av Papervision 2.0. I den nya versionen fanns färska effekter så som skuggning, frustrum-utgallring, Collada animationer och användning av flera ViewPort3D klasser i samma applikation. (Allen et al 2008 s. 291 f.)

Strax efter lanseringen av den nya versionen anslöt sig Andy Zupko till kärnutvecklargruppen. Han bidrog bl.a. med olika effekter till Papervision. Vidare grundades en hjälputvecklar grupp som tog hand om att utveckla icke-kärnrelaterade frågor. (Allen et al. 2008 s. 291 f.)

2.4 Konkurrenter

Det finns flera motsvarande system som Papervision men ofta blir dessa projekt ändå på hälft. Orsakerna är många, men den vanligaste är att tiden inte räcker till att utveckla programmet eller att projektet faller bakom något annat liknande system som utvecklas snabbare. Ibland kan det också hända att projektet från början har gjorts så invecklat att det är svårt att utvidga projektet.

I detta kapitel är några av de populäraste konkurrenter till Papervision presenterade. Dessa system kräver inget annat av användaren än Flash Player för att köras. Vanligen är Flash Palyer redan installerat i webbläsaren.

2.4.1 Away3D

Away3D är ett projekt byggt på öppen källkod och licenserat under Apache License 2.0. Away3D liknar mycket Papervision3D eftersom den i sitt ursprung var en förgrening av Papervision 1.0. Det var Alexander Zadorozhny och Rob Bateman som började utveckla Papervision för sina egna behov och då föddes Away3D. Senare, då Bateman och Zadorozhny hade utvecklat klart jämfördes Away3D och Papervision3D sinsemellan. Papervisions utvecklare Carlos Ulloa och Ralph Hauwert var övertygade och ville implementera funktioner från Away till Papervision, som sedan lanserades som version 2.0. (Grden 2007)

Away3D utvecklas av ett tiotal programmerare. Det har ett väl dokumenterad API vilket gör projektet till ett mycket attraktivt alternativ till Papervision3D. Away3D används redan i större kommersiella projekt. Koden är skriven direkt på ActionScript 3 och den utnyttjar redan Flash Player 10:ans egenskaper, vilket gör motorn mycket effektiv.(Kallonen 2010 s. 5, Zadorozhny 2007)

2.4.2 Sandy 3D

Sandy 3D är ett projekt byggt på öppen källkod och som utvecklas av ett litet team på cirka fem programmerare. Sandy finns tillgänglig i tre versioner: ActionScript 2, ActionScript 3 och haXe. Den stöder endast Flash Player 9 och tidigare versioner. Det betyder att den inte iakttar användarmaskinens hårdvara. Däremot stöder den Flash Player 7, som ännu förekommer i äldre webbläsare. API:n är mycket bra dokumenterad, vilket gör det lätt att användas. Sandy 3D liknar mycket Papervision3D, då båda har samma egenskaper och möjligheter. (Sandy 2009)

Sandy har dock hamnat aningen i skuggan av Papervision3D och Away3D. Den har inte ännu slagit igenom till de kommersiella projekten. Det är också oklart hur dess framtid ser ut. (Kallonen 2010 s. 5)

2.4.3 Sophie 3D

Sophie 3D är, till skillnad från tidigare nämnda projekt, en motor som fokuserar endast på att rendera ett objekt i taget. Den stöder bl.a. texturer, skuggor, genomskinliga material och oändligt många ljuskällor. Eftersom motorn är optimerad att rendera endast ett objekt kan man åstadkomma visuellt fantastiska effekter då all effekt kan koncentreras till objektet. (Sophie 2009)

För att köra programmet skall man ladda ner ett gratis paket till Flash. Det innehåller några exempel program, från vilka man kan lätt utgå då man skall programmera sitt eget projekt. Gratisversionen är endast för privat bruk och får inte användas i kommersiella projekt. En logotyp syns i övre vänstra hörnet som man inte får modifiera eller gömma. Vill man ha en version utan logotyp kan man köpa en Pro version för ca 300 € och då har man även rätt att använda programmet i kommersiella projekt. Programmet har dock inte öppen källkod. (Sophie 2009)

Programmet stöder importering av Wavefront Technologies utvecklade OBJ-format. De flesta 3D modelleringsprogram stöder detta format. Då man infört sitt 3D objekt kan man med Sophie ställa in parametrarna enligt eget tycke. Förutom OBJ-format stöder Sophie 3D även import av Collada format. Med Sophie 3D Pro versionen kommer en Sophie 3D Compressor, vilken ger möjligheten att komprimera filen med 75 %. Detta leder till snabbare laddningstid och mjukare interaktion. (Sophie 2009, Wavefront 2010)

2.4.4 Alternativa3D

Alternativa3D har en bakgrund på över tio år och har varit framgångsrik. Det har ursprungligen programmerats av ett ryskt team. Programmet är proprietär, d.v.s. programmet har stängd källkod. Det är gratis att använda Alternativa3D, men man är tvungen att lägga in en synlig logo eller text på Alternativa3D. Det går inte att köpa en version utan logotyp. Versionen man kan ladda ner är inte den nyaste, utan den tidigare versionen. Den nyaste versionen är endast för Alternativa3D:s eget samt deras samarbetspartners bruk. Däremot är det fritt att använda programmet privat likaså kommersiellt. Man kan ladda ner ett SWC bibliotek som lämpar sig för både Flash och Flex. (Alternativa 2010)

Flash Player 10:ans egenskaper har utnyttjats redan i ett tidigt skede. Utvecklarteamet har också lagt mycket tid på att bygga ut en server så att flera deltagare skall kunna spela samtidigt. Då man kombinerar dessa egenskaper skapar de en otroligt kraftfull motor, med vilken man kan åstadkomma väldigt fina resultat. (Alternativa 2010)

Alternativa3D är mest känd för att deras motor passar bäst för spel. Man kan bl.a. spela spel utvecklade av Alternativa3D på sociala medier, så som Facebook. Alternativa3D:s framtid ser ut att fortsätta konkurrenskraftigt inom spelbranschen. (Alternativa 2010)

2.5 Framtid

Detta kapitel handlar om hur jag ser framtiden för Papervision. Dessa visioner grundar sig på min egen erfarenhet samt egna tankar och diskussioner med andra programmerare.

Eftersom den tidigare versionen, Papervision 2.0:s hierarki inte kunde dra nytta av Flash Player 10:ans egenskaper började utvecklandet av PapervisionX eller senare kallad Papervision 3.0. För att få nyttan av dessa nya egenskaper måste man bygga om hela kärnstrukturen från grunden. Efter att Papervision 3.0 skulle vara färdig var det meningen att implementera vissa nyheter till den äldre Papervision 2.0. (PapervisionX 2009) Detta utfördes dock aldrig.

I september 2009 meddelade Ralph Hauwert att han lämnar Papervision teamet för att koncentrera sig på andra saker. Det var han och Tim Knip som utvecklade Papervision 3.0 men eftersom den inte var i ett stabilt läge kunde man inte publicera den. Arbetet som de gjort skulle de hålla för sig själva, vilket ledde till att den förväntade Papervision 3.0 aldrig kom ut. Sedan dess har det inte utvecklas på Papervision3D. Det har ändå inte till dags dato meddelats officiellt att Papervision3D skulle vara avslutat.

Papervision3D lever fortfarande genom den tidigare versionen som är stabil och fullt fungerande. Från och med juli 2010 kan man se exempelvideon på hur man bygger Papervision3D applikationer på webbsidan Lynda.com. (Papervision 2010a)

Det finns dock alltid risker i att programmera på en plattform som inte längre utvecklas. Råkar man på en bugg eller ett problem som ligger i kärnkoden finns det inte mycket som man kan göra. Då datorerna och resten av webben utvecklas konstant tror jag personligen inte att Papervision kommer att hänga med länge då den blir föråldrad. Programmerare förflyttar sig till andra plattformer som Away3D och Alternativa3D, vilka båda stöder den nyaste tekniken.

3 PAPERVISION3D EGENSKAPER

3.1 Initiera Papervision

I detta kapitel behandlas kort de olika versioner Papervision3D som finns tillgänglig samt installering och användning av dem. Fastän Papervision3D är ett klassbibliotek och inte en mjukvara går programmet bekvämt att installeras till Adobe Flash eller Flash Builder för en mjukare och bekvämare användning.

3.1.1 Versioner

Papervision3D finns tillgänglig i två olika versioner; ActionScript2 samt ActionScript3. Den nyaste, ActionScript3 versionen, är rekommenderad att användas eftersom den är snabbare, speciellt då den skall räkna ut komplicerade vyer. ActionScript3 versionen är stabilare och innehåller mer funktioner och effekter än ActionScript2 versionen. Utvecklandet av ActionScript2 versionen är avslutad. (Papervision 2010c)

ActionScript2 versionen kan användas vid ett sådant tillfälle där ActionScript3 versionen inte kan utnyttjas. En sådan situation kunde vara då ett projekt skall uppdateras eller ifall kunden har som önskemål att använda ActionScript2. ActionScript2 versionen fungerar i Flash Player8. (Papervision 2010c)

Papervision3D:s nyaste version är 2.1. Den tidigare uppdateringen 2.0 lanserades med kodnamnet ”Great White”. Version 2.0 hade stora ändringar i kärnkoden vilket gjorde

motorn snabbare och hämtade nya effekter och egenskaper inklusive import av Colladamodeller. (Papervision 2010c)

3.1.2 Installering och användning

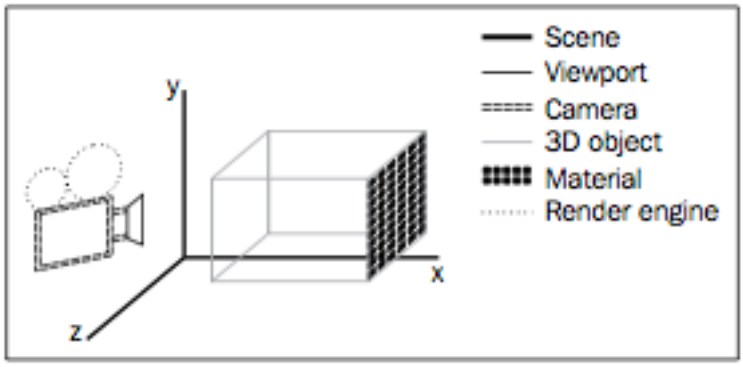
Papervision3D källkoden laddas kan ner i två olika format; kompilerad SWC-fil och okompilerad Zip-fil. Zip-filen innehåller källkoden i ett format som kan öppnas och läsas. Källkoden är väl kommenterad och man kan ha nytta av att se förklarningarna. I SWC-filen kan användaren inte läsa källkoden, utan klasserna är tillgängliga endast för datorn att läsa.

Efter nerladdad källkod sparas filen på datorn på önskad plats, men det är rekommenderat att spara filen i roten av samma filstruktur som själva projektet ligger. Därefter anger man filstigen i Flash, där källkoden är sparad. Ifall projektet flyttas, rekommenderas att ge filstigen relativ för att källkoden skall vara igenkänd. (Winder & Tondeur 2009 s. 8 ff.)

Med Zip-filen medföljer en dokumentation om alla Papervision3D klasser. Dokumentationen är en HTML (Hyper Text Markup Language) sida och är genererad för ASDocsverktyget i Flash Builder. Dokumentationen är ett bra verktyg som underlättar arbetet. Användning endast av dokumentationen kan dock vara för lite. Det finns flera exempelfilmer med källkod på internet, som man kan ha som stöd vid programmering av Papaervision3D projekt. (Winder & Tondeur 2009 s. 8 ff.)

3.2 Papervision3D huvudklasser

Papervision är byggd på ett bibliotek av olika klasser, där alla har sin specifika uppgift. Genom att använda dessa klasser i olika situationer kan man skapa olika resultat. I detta kapitel behandlas huvudklasserna i Papervision3D. Dessa kan visualiseras enligt Figur 1.



Figur 1. Visualisering över de essentiella delarna i Papervision3D enligt Winder & Tondeur (2009 s. 38)

3.2.1 Scene3D

Scenen är var alla objekt är placerade. Den innehåller hela 3D miljön och behandlar alla objekt som skall renderas på scenen. Scene3D klassen utvidgar DisplayObjectContainer3D klassen. Man kunde tänka scenen som stoff kontainer för projektet. (Lively 2010 s. 38-54)

I själva verket ser man aldrig materialet i scenen. Det behandlas matematiskt och när det har renderats, adderas materialet till viewporten. Detta är ca en tredjedel av arbetet på processorn. De två övriga tredjedelar går till rendering och projicering till viewport. (Lively 2010 s. 38-54)

3.2.2 Viewport3D

Viewport3D klassen har 4 parametrar. De två första är höjd och bredd. Dessa är mycket viktiga, då det gäller att optimera projektet. Ju större viewport desto mer processorkraft behövs. Använder man värden 0,0 renderas projektet enligt Flash-presentationens storlek. Ändringar i viewport storleken ändrar inte på objektens storlek. De två övriga parametrarna bestämmer ifall viewporten skall skalas automatiskt till flash storleken. Den sista parametern bestämmer ifall man skall addera interaktivitet. Denna lägger lyssnare på objekten via InteractiveSceneManager-klassen. (Lively 2010 s. 52 f.)

Det är möjligt att göra flera instanser av Viewport3D där man använder samma Scene3D- och Camera3D-klasser och samma eller olika renderings motorer. Möjligheterna

är många. Den stora fördelen med detta är att man undviker olika sorterings problem. (Lively 2010 s. 52 f.)

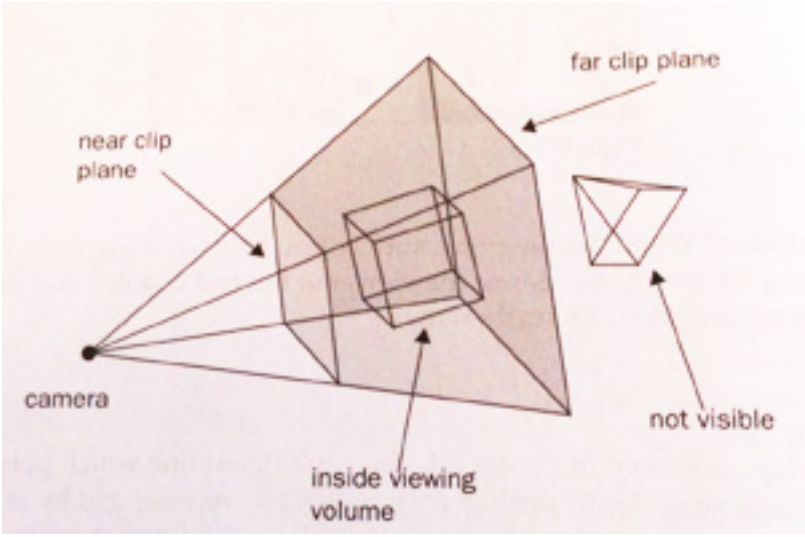
BitmapViewport3D är en utvidgad klass till Viewport3D. Med denna klass kan man bl.a. hämta bitmap data. Filter, som suddighet, kan köras direkt till klassens instans. (Kallonen 2010 s. 32)

3.2.3 Camera3D

Camera3D är en klass som ärver egenskaper från DisplayObject3D klassen. Vilket betyder att man kan bl.a. röra den i x-, y-, och z-koordinatsystemet. Man kan även rotera kameran på samma sätt som vilken annan instans av DisplayObject3D. Kameran har inte en visuell representation på scenen, utan enbart en virtuell punkt från vilken vi kan se vad som renderas på scenen. Man kan man ändå manipulera kamerans egenskaper mycket effektivt. Man kunde tänka att kameran är ögonen för projektet. Grundegenskaper är:

- Focus
- Zoom
- Field of view
- Near plane och Far plane

Kamerans fokusvärde är samma som near plane, dvs framplan. Frustum innehåller allt som är synligt på scenen. I Figur 2 är frustrum visulaiserat som ”inside viewing volume”. Genom att ändra på fokus värdet flyttar man på framplanet. Det som faller utanför frustum renderas inte och är inte synliga. Se Figur 2. (Winder & Tondeur 2009 s. 129-135)



Figur 2. Visualisering över kamera och dess begränsningar (Lively 2010 s.38).

Högre fokusvärde ökar på distansen mellan kameran och framplanet. Objekt som är nära och längre bort ser ut som att vara nära varandra, även om distansen är stor mellan dem. Förminskar man fokusvärdet förstorar man synfältet. Detta är som att använda en vidvinkel lins, där nära objekt förstoras och objekt längre bort förminskas. Minskar man alltför mycket orsakas distorsion. (Winder & Tondeur 2009 s. 129-135)

Genom att öka på zoomvärdet kan man förstora objektens storlek på scenen. Ändringar i zoomvärdet upplevs lika på scenen som ändringar i fokus värdet. Zoom funktionen ändrar dock inte på frustum. Rätt användning av zoom och fokus är viktigt för att det inte ska uppstå distorsion. (Winder & Tondeur 2009 s. 129-135, Kallonen 2010 s. 32 f.)

Förutom att flytta på kameran i olika riktningar kan man även bestämma ett objekt som kameran alltid följer. Kameran kan följa vilken DisplayObject3D-instans som helst. Detta sker via target funktionen. Med orbit funktionen kan man bestämma ett fast avstånd till objektet och kameran cirkulerar på denna radie runt objektet oberoende av kamerans placering.(Winder & Tondeur 2009 s. 129 -135)

Det finns flera utvidgade klasser av Camera3D; Free, Spring och Debug. Typen Free är standard och är vanligen använd. Spring kamera följer ett objekt på avstånd. Kameran reagerar med ett litet dröjsmål. Man kan manipulera de vanligaste fysikaliska egenska-

per, som massa, friktion och dröjning. Med dessa egenskaper kan man skapa en mycket realistisk känsla, där kameran inte exakt följer objektet.

Debug kameran är meningen att användas vid projektets utveckling. Då man skall hitta en bra vinkel och rätta inställningar är Debug kameran bra. Genom keyboard knapparna eller musen kan man ändra på värden medan applikationen är i gång. Debug kameran är inte meningen att användas i slutliga projektet. Vill man ha kamerans interaktion, kan man kopiera den från klassen eller bygga om själv.(Winder & Tondeur 2009 s. 129-135, Kallonen 2010 s. 32 f.)

3.2.4 Rendering

Renderingsmotorn utför all nödvändig kalkylering för att rita 3D data. Den plockar att data från Scene3D och ritar ut det som syns enligt Camera3D ögon. Denna process är den mest krävande och kräver hög processeringseffekt. Man kunde tänka sig rendering som projektets artist. Det finns tre olika renderingsmotorer eller klasser: BasicRendererEngine, LazyRendererEngine och QuadrantRendererEngine. (Lively 2010 s. 40-54)

BasicRenderEngine är den vanligaste motorn och lämpar sig till de flesta projekten. Motorn är snabb då den använder ett simpelt sätt att renderera objektens djup och placering. För att modifiera renderingsegenskaper kan man använda filter. Det finns dock bara ett filter; FogFilter. Filtret fungerar genom att generera enkla färgade fyllningslager. Dessa lager orsakar att objekten ”försvinner” då avståndet till kameran ökar. (Kallonen 2010 s. 34 ff., Zupko 2008a)

LazyRenderEngine är en utvidgad klass till BasicRendererEngine. Dess egenskaper är lika som BasicRendererEngine och hämtar inte några extra egenskaper. Användningen av motorn sker genom att deklarera Scene3D-, Camera3D- och Viewport3D-klasserna. Källkoden blir kortare och tydligare. (Kallonen 2010 s. 34 ff.)

Behöver man en mera utvecklad motor för att renderera problemsituationer kan man använda QuadrantRenderEngine. Denna motor har två olika filter. Med QaudrantZFilter kan man lösa sorteringsproblem som skulle uppstå ifall man använde BasicRenderEngi-

ne. QuadrantFilter löser problem där objekt går igenom varandra. Filtret splittrar objekten vid skärningspunkten. Detta gör det möjligt att renderera objekten utan problem. Det är dock mycket krävande att räkna problemsituationer för hela skärmen. QuadrantRenderEngine delar skärmen adaptivt till rektanglar där det är mycket lättare att lösa problemsituationerna. Det går att bestämma i hur många rektanglar skärmen delas in i. (Kallonen 2010 s. 34 ff.)

Man skall dock fundera noga ifall det är värt att använda QuadrantRenderEngine, eftersom den är mycket mer processorkrävande. Ibland kan man lösa problem t.ex. genom att använda flera lager av Viewport3D klassen. Slutligen handlar det om att optimera sitt projekt. (Lively 2010 s. 40-54)

3.2.5 DisplayObject3D

DisplayObject3D motsvarar MovieClip från Flash. Alla objekt placerade på scenen i ett Papervision3D projekt är visningsobjekt. DisplayObject3D-klassen representerar instanser av 3D objekt. Dessa inkluderar inte bara alla objekt som renderas på scenen utan också kamera och dess mål. (Lively 2010 s. 41 f.)

DisplayObject3D-klassen stöder de grundliga funktionaliteter såsom positionering i x-, y- och z- koordinatsystem samt rotation och gradering i alla tre koordinataxlars riktning. (Lively 2010 s. 41 f.)

DisplayObject3D-klassen är inte en abstrakt grundklass, vilket ger möjligheten att skapa tomma DisplayObject3D objekt som kan fungera som behållare för andra 3D objekt. Fysiken; massan, farten, accelerationen och friktion adderas till DisplayObject3D-klassen. Man kunde tänka att klassen fungerar som kropp för projektet. (Lively 2010 s. 41 f.)

3.3 BasicView

BasicView är en klass där de grundliga Papervision3D variablerna är automatiskt deklarerade. Dessa är alltså: Viewport3D, Camera3D, Scene3D och BasicRenderEngine.

Modifiering av standardinställningar är möjligt i efterhand i applikationen. BasicView är skapad för att förenkla och snabba deklaration av Papervision3D applikationer. Den fungerar ypperligt i mindre komplicerade projekt. (Lively 2010 s. 67-73)

Renderering sker automatiskt genom startRendering funktionen. Denna funktion körs konstant och man behöver endast starta den. StartRender funktionen kör onRenderTick funktionen. Förövrigt är det möjligt att renderera på två andra sätt. SingleRender funktionen renderar en bild på scenen. Till SingleRender behöver man inte mata parametrar, utan den tar dem automatiskt. Vill man renderera scenen genom t.ex. en annan kamera är det möjligt med renderScene funktionen. Vill man ändra vad render funktionen gör kan man överlappa onRenderTick funktionen och modifiera funktionen enligt eget behov. (Lively 2010 s. 67-73)

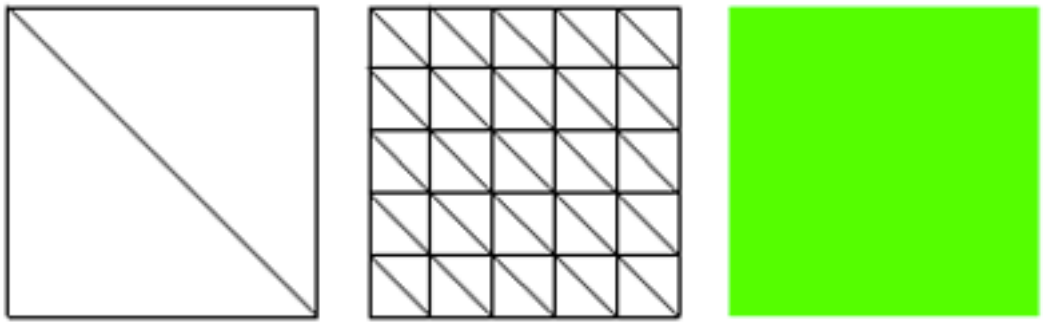
BasicView har en utvidgad ReflectionView-klass. Genom att använda denna klass kan man skapa reflektioner. Detta är ett mycket enkelt sätt att skapa extra visuella effekter. Att foga in reflektioner kan dock vara mycket effektkrävande och kan göra applikationen trög. (Lively 2010 67-73)

3.4 Primitiver

I detta kapitel behandlas de vanligaste primitiver, dvs geometriska figurer. Förutom dessa finns det också några intressanta extra former som pilar och pappersflygplan. Dessutom är det möjligt att forma egna primitiver genom diverse geometriska räkningar. (Lively 2010 s. 76 f.)

Användning av primitiver är ofta grunden till flera Papervision projekt. Grundligen består primitiverna av polygoner av olika mängd. Vid skapandet av en primitiv är det möjligt att bestämma i hur många segment, polygoner, figuren är uppdelad i. Ju flera segment desto mjukare utseende, men också mer effektkrävande. Enheterna är Papervision enheter. (Lively 2010 s. 76 f.)

3.4.1 Plan



Figur 3. Exempel på olika plan primitiver.

Plan, engelska *plane,* är en platt yta som sträcker sig i alla riktningar, samlad av en mängd punkter eller linjer. Ett plan har två dimensioner; höjd och bredd. Ett plan har ingen tjocklek. (Plane 2011)

Planet hör till de mest använda primitiverna i Papervision3D, eftersom det är lätt att räkna ut hur de skall ritas. Genom att texturera plan på olika sätt kan man enkelt skapa fina visuella applikationer såsom bildkaruseller.

Ett plan i Papervision har höjd-, bredd-, segment- samt materialparametrarna. Alla dessa är modifierbara i konstruktorn. (Papervision 2010b)

3.4.2 Kub

Figur 4. Exempel på olika kub primitiver.

I Papervision3D är kub, engelska *cube,* en primitiv. Det betyder dock inte en rätvinklig tredimensionell figur enligt geometrisk definition. Kuben i Papervision3D kan ha olika

bredd, höjd och djup. I praktiken betyder det att kub är en tredimensionell fyrkant med åtta hörn. (Cube 2011)

Då man definierar en kub skall man ge material eller textur för sidorna. Materialet kan vara samma eller olika för respektive sida genom att använda MaterialsList-klassen. Sidornas bredd, längd och höjd samt segmenten för respektive sidor skall även anges vid definiering av kuben. (Papervision 2010b)

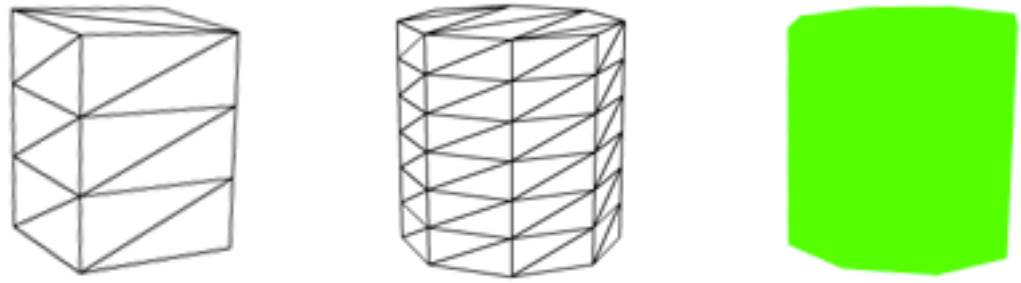
3.4.3 Sfär

Figur 5. Exempel på olika sfär primitiver.

Sfär, engelska *sphere,* är en klotformad symmetrisk kropp. Alla punkter på sfären ligger på samma avstånd från figurens centrum. (Sfär 2011)

En sfär i Papervision har radie, antal segment i höjd och bredd, antal segment i x- och y-led samt minsta antalet segment i respektive lägen. Antalet segment bestämmer hur mjuk figuren ser ut. Ibland kan dock för många segment orsaka försämring av texturen. Vid polarområden, där segmenten blir så små, ser man exempel där texturen kan försämras. (Papervision 2010b, Kallonen 2010 s. 38)

3.4.4 Cylinder

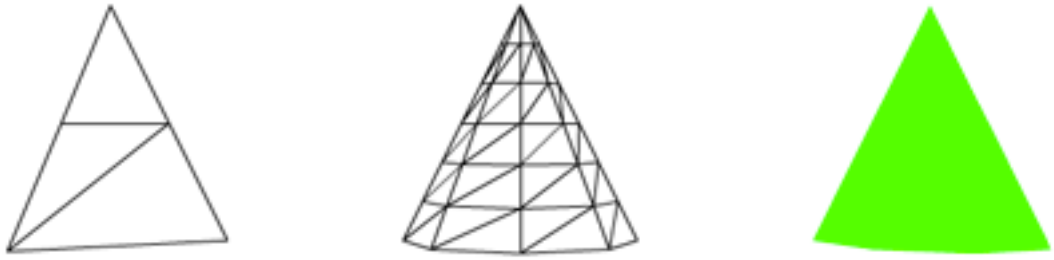


Figur 6. Exempel på olika cylinder primitiver.

Cylinder, engelska *cylinder,* är en tredimensionell rymdgeometrisk kropp som avgränsas av två likformiga plana ytor och den mantelyta som löper längs basytornas hela omkrets. Basytorna bör vara lika vinklade och får inte befinna sig på samma punkt. (Cylinder 2011)

I Papervision3D är cylinder definierad med textur, höjd, radie, antal segment, takets radie samt genomskinlighet i basplan. (Papervision 2010b)

3.4.5 Kon



Figur 7. Exempel på olika kon primitiver.

Kon, engelska *cone,* är en geometrisk kropp som bildas av linjer mellan samtliga punkter på konturen av en plan figur (basytan) och en punkt utanför planet (spetsen). Är basytan en cirkel fås en cirkulär kon; är den en polygon fås en pyramid. (Kon 2011)

I Papervision3D definieras en kon med material, basens radie, höjd samt antal segment i höjd och bredd. (Papervision 2010b)

3.5 Material

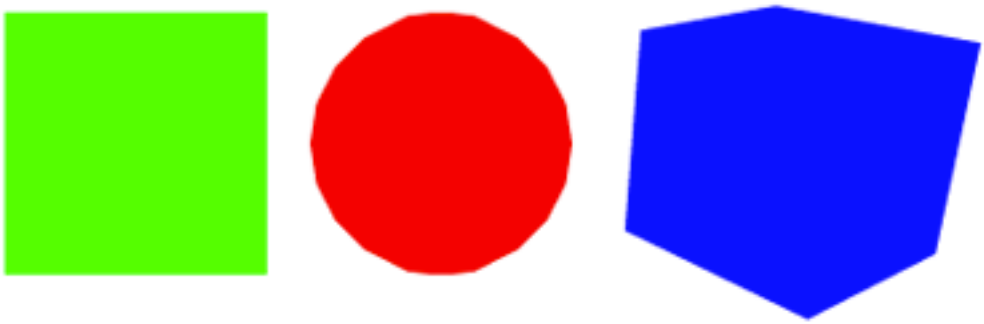
Genom att lägga till material till 3D objekt skapar man själva synligheten. Utan material är objekten genomskinliga, alltså osynliga. Det finns flera olika material man kan använda, även olika skuggor eller ljuskällor kan användas för att skapa en mer realistisk bild. Man kunde tänka material som kläder för objekten. Utan material kan man inte lägga interaktivitet på objekten. (Lively 2010 s. 127)

Detta kapitel redogör för de vanligaste typer av material. Eftersom materialet består av MaterialObject3D klassen har de följande egenskaper gemensamma.

- Dubbelsidighet (double-sided)
- Motsats (opposite)
- Slät (smooth)

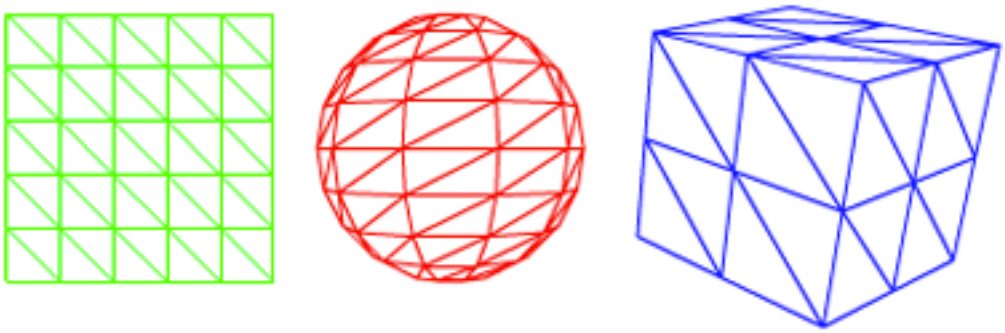
Med dubbelsidighet kan man bestämma om samma material skall synas på andra sidan av objektet, oftast plan. Har man till exempel en sfär och vill visa materialet på insidan av sfären använder man motsats egenskapen. Vid transformering och skalning av 3D objekt kan materialet bli av dålig kvalitet. Detta kan förbättras genom släthetsegenskaper. Användning av släthet kan vara bra men är effektkrävande. (Lively 2010 s. 127, Winder & Tondeur 2009 s. 83 ff.)

3.5.1 Färgmaterial



Figur 8. Exempel på olika färgmaterial.

Ett av de grundläggande material som används för att fylla ett objekt är färg. I Papervision3D kallas detta för ColorMaterial. ColorMaterial fyller objektet med en färg utan toning. ColorMaterial är lätt och enkel att använda. Till färgmaterialet anges färgen i hexadecimal form samt genomskinlighetsgraden av färgen. (Winder & Tondeur 2009 s. 83 ff.)



Figur 9. Exempel på olika Wireframe material

WireframeMaterial är fyllnadsmaterial lika som ColorMaterial, men WireframeMaterial fyller inte hela objektet utan endast ritar ut linjer där segmenten ligger. WireframeMaterial är standard material för de flesta primitiverna. Användning av WireframeMaterial vid interaktivitet är inte rekommenderat, eftersom interaktiviteten endast gäller där linjer är ritade. (Winder & Tondeur 2009 s.83 ff.)

3.5.2 Bitmapmaterial

BitmapMaterial klassen skapar material, består av bitmapdata. Denna klass finns för att skapa mera detaljer och intresse till 3D objekten. Bitmapmaterialen har motsvarigheter från de grundläggande materialen. Dessa är BitmapColorMaterial och BitmapWireFrameMaterial samt BitmapMaterial och BitmapAssetMaterial. (Winder & Tondeur 2009 s.83 ff., Kallonen 2010 s. 41)

Till Bitmapmaterialens egenskaper hör bland annat att man kan direkt tillägga filter såsom oskärpa och glöd. (Kallonen 2010 s. 41)

3.5.3 Filmmaterial

Det finns två olika sätt att tillämpa film som material till ett objekt; MovieMaterial och MovieAssetMaterial. MovieMaterial skapar texturen av en definierad MovieClip. Movie Material kan vara animerad och genomskinlig. Det är också möjligt att definiera ett litet fyrkantigt område från en MovieClipen som används för material. MovieMaterial ger goda möjligheter för interaktivitet. De använda MovieClip kan ha egna lyssnare som möjliggör t.ex. musklick för att köra en funktion. (Kallonen 2010 s.41)

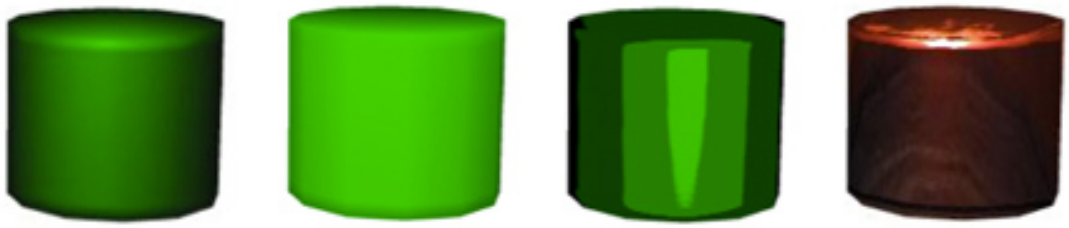
MovieAssetMaterial är lika som MovieMaterial förutom att man använder material som finns i Flash biblioteket. Materialet som används via biblioteket skall ha registreringspunkten uppe i vänstra hörnet för att Papervision3D skall registrera materialet på rätt sätt. (Papervision 2010b)

3.5.4 Skuggor och skuggning

För att skapa skuggor bör man ha en ljuskälla. I Papervision3D finns det en ljuskälla; PointLight3D-klassen. Klassen är utvidgad från DiplayObjekt3D klassen vilket ger möjligheten att t.ex. röra ljuskällan fritt i 3D-rymden. Någon egentlig klass för skuggor finns inte i Papervision3D. Effekt ansvariga Andy Zupko har ändå skrivit en klass där skuggan räknas väldigt noga på ett 3D objekt. Klassen heter ShadowCaster och kan laddas från Zupkos blogg. ShadowCaster klassen är fullt kompatibel med Papervision3D men är inte en standard klass och bör därför laddas separat. (Papervision 2010b, Zupko 2008b)

Figur 10. Exempel på användning av ShadowCaster-klassen (Zupko 2008b).

Däremot finns det flera olika skuggningar i Papervision3D. Detta betyder att istället för en liktonad färg på ett objekt kan man simulera olika toningar på objekten. Skuggning kallas för *Shaders*. Användning av shaders är mycket effektkrävande, men också visuellt mera verklighetstroget. I Papervision3D finns fem olika skuggningar; FlatShader, CellShader, GouraudShader, PhongShader och EnvMapShader. (Papervision 2010)



Figur 11. Exempel på Flat-, Cell-, Gouraud-, Phong-, och EnvMapShader (Kallonen 2010).

3.6 Interaktivitet

Det finns flera olika sätt att addera interaktivitet i Papervision3D. För att skapa interaktivitet används två klasser för detta; InteractiveSceneManager och VirtualMouse. (Papervision 2010)

Eftersom allt som renderas konverteras till bitmapdata och inte enskilda MovieClip kan man inte addera interaktivitet direkt på objektet. InteractiveSceneManager klassen lyssnar på musen och simulerar situationen som om det vore vanliga MovieClips på scenen.

Från InteractiveSceneManager klassen hittas de flesta MouseEvent funktionerna som också finns normalt i Flash. (Papervision 2010, Kallonen 2010 s. 49 f.)

Då materialet på objekten är en MovieClip och musen klickas, känner InteravtiveSceneManager händelsen och tillsammans med VirtualMouse klassen körs MovieClippens egna händelse, såsom MouseOVER och MouseCLICK. (Papervision 2010, Kallonen 2010 s. 49 f.)

Det är även möjligt att addera interaktivitet genom att lägga lyssnare direkt på Viewport3D instansen. Då registreras alla lyssningar på den renderade bilden. Funktionen fungerar lika som om man har en lyssnare på en vanlig MovieClip och har alla samma egenskaper. Användning av denna metod fungerar endast då det finns ett objekt på scenen eller om samma sak skall hända till alla objekt. Man skall även lägga Viewport3D och materialets *interactive* egenskap till sann. (Papervision 2010, Kallonen 2010 s. 49 f.)

4 KRAVSPECIFIKATION

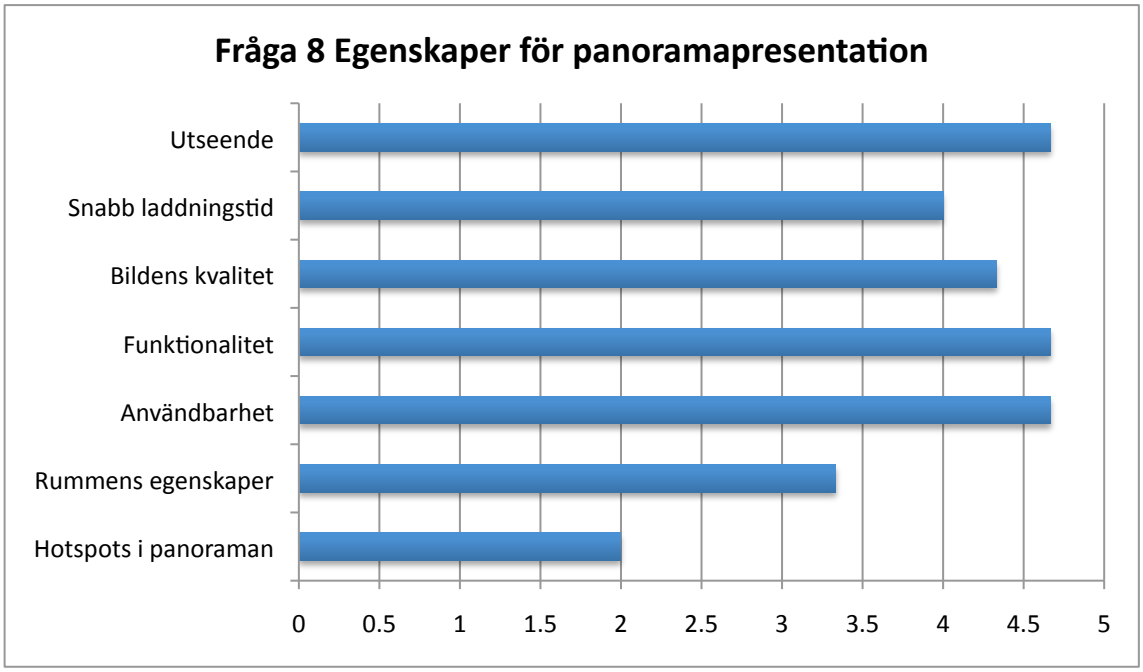
4.1 Kravinsamling

Kravspecifikationen är en mycket viktig del av programmet när det gäller beställningsarbeten. För att ta reda på vilka kraven är, har i detta examensarbete utnyttjats intervju metoden. Därefter har teorin prövats med eye tracker. Resultatet av eye tracker kunde också användas som en del av kravinsamling.

4.1.1 Intervjuer

Målsättningen med intervjuerna var att få experters synvinklar inom deras eget område. I intervjuerna deltog fem personer varav en expert inom panorama uppbyggnad, två experter inom webbutveckling samt en expert inom resurshantering och en expert inom marknadsföring. Informanterna representerade båda könen i åldern kring 35. Majoriteten av intervjuerna genomfördes i Arcada. Intervjufrågorna finns i Bilaga 1.

Fråga 1 hade som mål att få veta lite bakgrundsinformation inom panoramapresentationer. Dessutom var jag ute efter en öppen inledning där informanten fritt fick själv berätta om sina erfarenheter. Den öppna stämningen skapade en mjuk öppning till intervjun. Fråga 2 var en följdfråga till den första frågan och lämnades obesvarad ifall svaret till den första frågan besvarade fråga 2. Frågorna 3 och 4 riktade sig mera mot hur innehållet skall struktureras på presentationen. Målet var att hitta den information som allmänt glöms bort och att ta reda på sådant som borde elimineras från informationen. Svaret till fråga 5 skulle bestämma hurdan navigering presentationen skulle få och riktade endast till ifrågavarande problem. Fråga 6 var en rankingfråga som hade som mål att hitta informantens, och industrins professionella personers, åsikter om olika egenskaper. Resultaten från frågan finns sammanställda i Figur 12.



Figur 12. Resultatredovisning över fråga 8.

Sista frågan var en öppen fråga för att väcka diskussion och låta den intervjuade presentera sådant som inte kommit upp i intervjun. Det var en bra avslutning på intervjun då det också kom fram övriga aspekter som inte jag tidigare tänkt på bland annat att virtuella presentationen kan ge mervärde vi marknadsföring för nya studenter.

Intervjun fortlöpte smidigt då jag hade träffat eller kände personerna från tidigare och kunde ställa följdfrågor som passade i situationen. Intervjuer visade sig vara en bra metod för att få en översikt om vilka krav och egenskaper som behövdes.

Anteckningar samt ljudbandning användes som dokumenteringsteknik, vilket visade sig fungera väl. Då jag inte alltid hade möjlighet att anteckna allt under intervjun, kunde svaret granskas från inspelningen.

4.2 Eye Tracker

Detta kapitel behandlar hur jag använt Eye Tracker för att granska hur olika experter ser på panoramapresentationen. En eye tracker apparat spelar in ögats rörelse och fokusering för en noggrann analys. Resultaten över eye tracking finns i Bilaga 3 på DVD och forskningsfrågorna finns i Bilaga 2.

4.2.1 Eye Tracking

I examensarbetet användes Tobii T120 Eye Tracker. Denna apparat är specialiserad på händelser på datorskärmen. Eye trackern kan åskådas i Figur 13. Eye trackern är integrerad i en 17-tums TFT skärm. Eye trackern kräver en värddator som kör mjukvaran medan den externa skärmen spårar ögats rörelse. T120 modellen har en noggrannhet på 0,5 grader och en frekvens på 120 Hz. Trackern anpassar till huvudets rörelse och fungerar även om testpersonen använder glasögon. Svänger testpersonen för mycket på huvudet eller tittar bort reagerar trackern nästan omedelbart och fortsätter spårandet då fokuseringen returnerar på skärmen igen. (Tobii T120 2010)

Figur 13. Tobii T120 Eye Tracker (Tobii T120 2010).

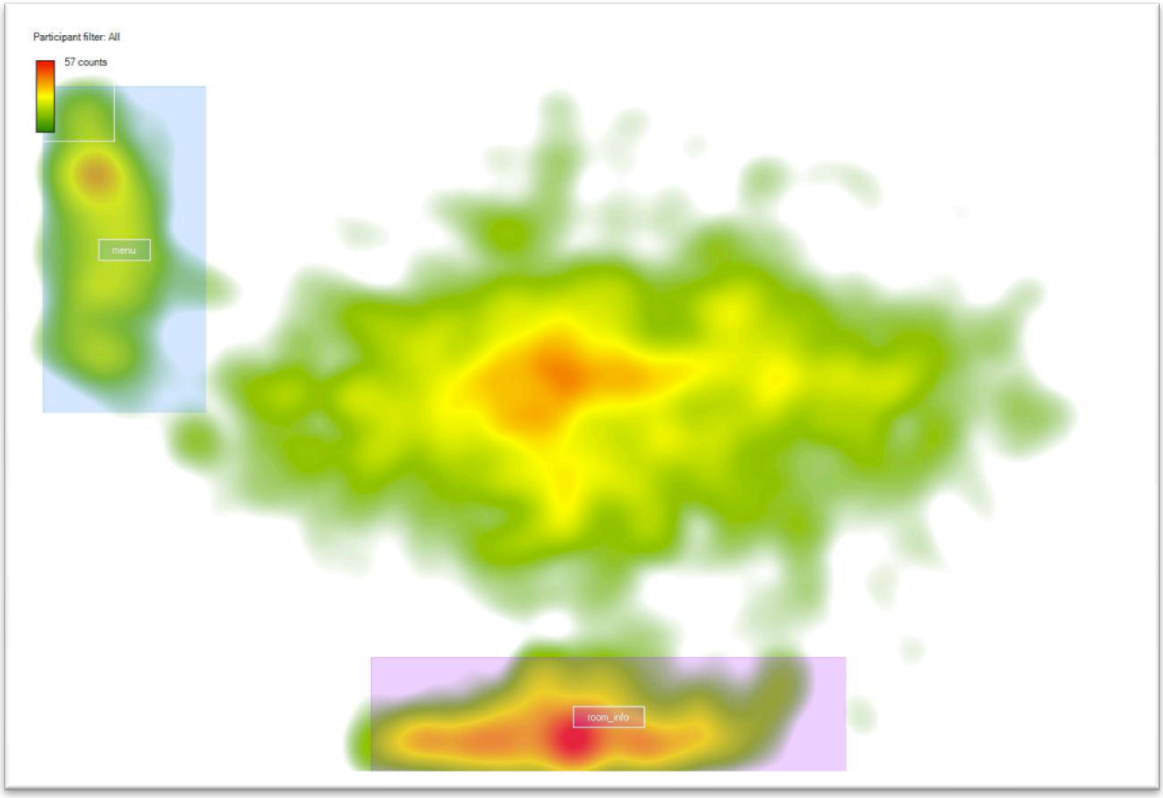
Målet med att använda eye tracking var att se skillnader mellan olika expertområden. I eye tracker testet deltog samma personer som i intervjun. Ingen av testpersonerna hade

tidigare haft kontakt med panoramapresentationen, men några var bekanta med motsvarande program.

För att resultatet skulle vara så naturligt som möjligt, fortlöpte testet öppet och testpersonerna fick ingen direkt uppgift att göra. Tidsbegränsningen var valfri då testpersonerna använde programmet tills de ansåg att det var nog och då avslutades också spårandet.

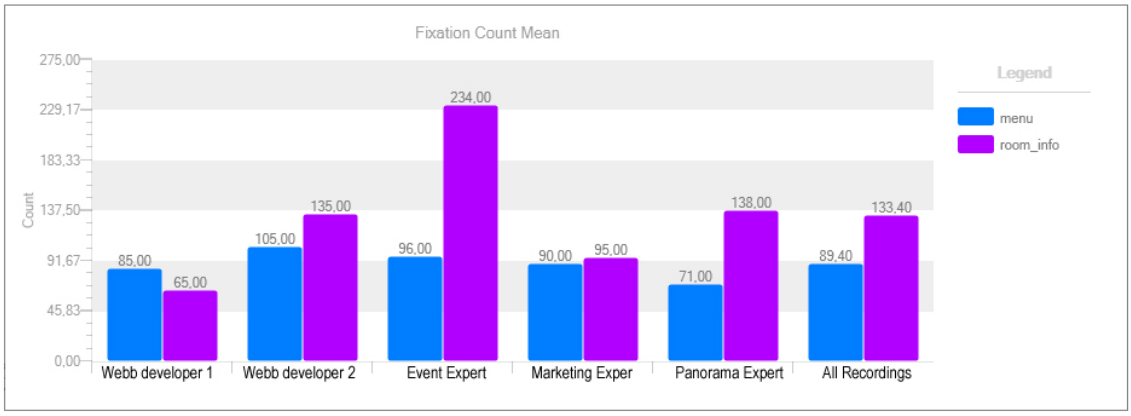
4.2.2 Resultat

Fråga 1 hade som mål att granska ifall instruktionerna fungerar i text form. Alla av deltagarna läste igenom texten, vilket antyder att det fungerade. Nästan alla stängde instruktionerna genast, endast en person fortsatte en stund innan denne gömde instruktionerna. Efter testet specificerades texten dock lite för att motsvara användargränssnittet bättre. Fråga 2 var en följdfråga till Fråga 1. Resultatet delades i två delar, varav 40 % läste rummets information, medan majoriteten roterade på panoraman. Frågorna 3 – 5 riktades till användargränssnittet och var användarna fokuserar mest på, Figur 14 visualiserar en värmekarta över de populäraste områden. Resultatet av Figur 14 bevisar att man inte lägger märke till hörnen eller kanter på presentationen. Då bildens kvalitet lite försämras vid dessa områden torde det inte ha betydelse då koncentrationen centreras på mitten av presentationen.

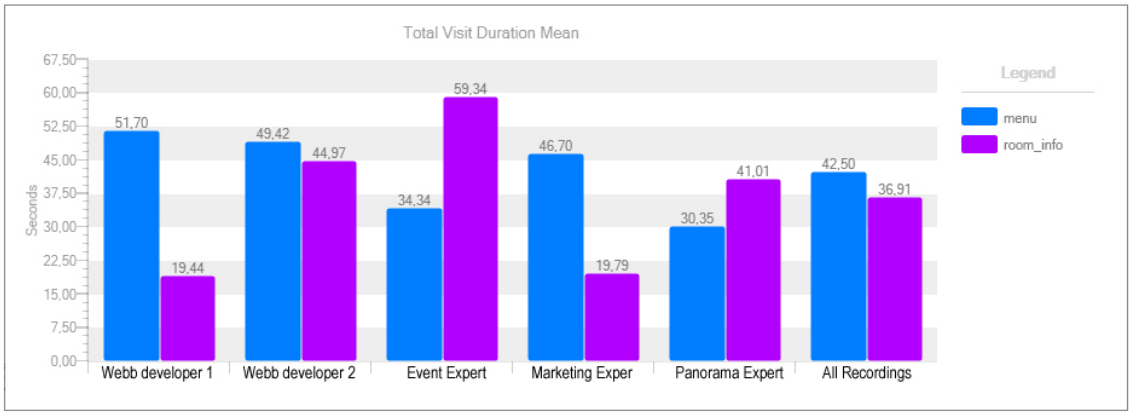


Figur 14. Värmekarta över de populäraste områden i presentationen.

Vid analysering av eye tracker data kan man även bestämma olika områden på skärmen för noggrannare analys. Vid analysering av presentationen valdes menyn till vänster och information fältet för närmare analys. Det intressantaste var att information fältet fick ett större antal preciseringar än menyn, medan den totala tiden var längre för menyn än vad information fältet fick. Skillnaden är visualiserad i Figur 15 och Figur 16. Detta betyder att informationen är noggrannare läst medan menyn har varit mera intresserande. Målet är att besökarna läser informationen som bevisat. Menyn är däremot också viktig för att besökarna skall hitta till de andra utrymmena. I Bilaga 4 finns övriga tabeller över information över de valda områdena.



Figur 15. Antal preciseringar.



Figur 16. Tid på besök.

4.2.3 Diskussion

Användning av eye tracking för analysering av hur olika personer ser på presentationen fungerade bra. Även om det i studien ingår endast ett få tal personer som deltagit i testet kan man hitta likheter. Saker som man redan vet, men inte har varit säker på kan bevisas med ett eye tracker test. I detta fall bevisades att koncentrationen centreras till mitten av bilden och kanterna blir utanför synen. Speciellt intressant var att konstatera att testpersonerna läste informationen även om de känner till utrymmena från tidigare.

Arbetandet med eye tracking kräver mycket tid och mängden data som man får efter ett test är enormt stort. Vid analyserandet av materialet följdes ingen teori, utan påståenden baserar sig på iakttagelser utgående från forskningsfrågorna.

Användning av eye tracker har varit mycket intressant och lärorikt. Det är ett mycket effektiv sätt att leta efter problem som uppkommer vid användning. Speciellt intressant är det att se hur olika personer använder samma applikation.

5 PRAKTISKA UTFÖRANDET

Den praktiska delen av detta examensarbete går ut på att bygga en virtuell presentation som presenterar de olika utrymmen i Arcada. Detta kapitel redogör för de centrala arbetsdelarna under projektet. Exempelkoden för funktionerna är förenklad för att gen en bättre helhetsbild. Fullständiga källkoden är bifogad i Bilaga 3.

5.1 Programmering

Då det gäller panoramapresentationer är det flera som undrar vad har det med 3D motorer att göra. Egentligen har det ganska lite med 3D motorers oändliga effekter och funktioner att göra, men det är grundfunktionen som jag utnyttjat i panoramapresentationen.

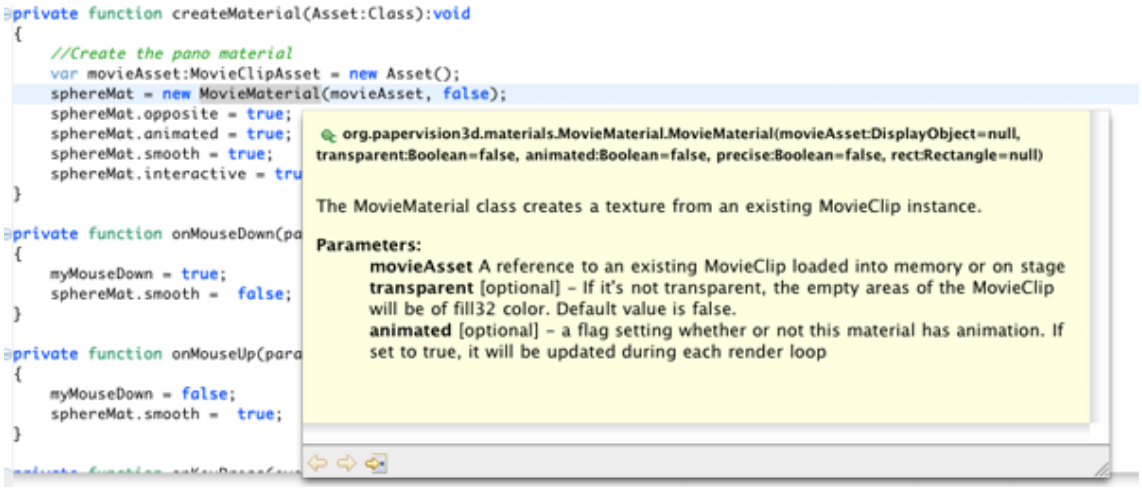
Dessutom är det ett mycket simpelt sätt att visa en panoramabild med en 3D motor. Först skapar man en sfär (eller annan primitiv) och sedan texturerar man sfären med panorama bilden. Slutligen lägger man en kamera i mitten av sfären och adderar allt på scenen. Därefter går det smidigt att bygga ut funktionaliteten.

Det finns flera olika primitiver som kan användas som underlag för panoramapresentation. I detta examensarbete har presentationen gjorts med en sfär. Det är också vanligt att använda en kub eller cylinder.

5.1.1 Arbetsverktyg

Jag valde att bygga presentationen med Adobe Flash Builder. Fördelar med att använda Flash Builder är dess funktionalitet. Med hjälp av den automatiska textinmatningen sparar programmeraren mycket tid och man behöver inte komma ihåg exakt hur t.ex. en klass är stavad. Dessutom har man stor nytta av att få klassens basinformation i ett litet

fönster (Figur 17).



Figur 17. Exempel på hur man får information i ett litet fönster i programmet Adobe Flash Builder.

Presentationen är byggd på en mxml fil där jag har inkluderat en ActionScript fil som innehåller all kod för initialisering och funktioner för Papervision3D. Mxml filen innehåller intagning av menyn samt de övriga bilderna.

5.1.2 Skapandet av presentationen

Jag valde att bygga presentationen på en sfär. Fördelar med detta är bl.a. att man kan texturera objektet med en gång. Detta sparar på applikationens storlek, men minskar också litet på bildens kvalitet. Jämfört med om man har en kub är man tvungen att ha sex olika texturer. Då kan man åstadkomma bättre kvalitet, men ökar betydligt på den slutliga applikationens storlek och därmed ökar också laddningstiden. Enligt intervjuerna och mina egna tankar förstärktes påståendet att kvaliteten på bilden inte är alltför noga. Detta kunde även bevisas via eye tracker testet.

Papervision3D variablerna initialiseras först. De innehåller skapandet av Viewport3D, Scene3D, Camera3D och renderings motor, som i detta fall användes av BasicRendererEngine. Initialiseringen ser ut på följande vis.

```
// Papervision Variables
viewport = new Viewport3D(1, 1, true, true);
pv3Canvas.addChild(viewport);
scene = new Scene3D();
camera = new Camera3D();
renderer = new BasicRenderEngine();
```

Sedan lägger man till en kamera och positionerar den i mitten av sfären. Detta gör man enkelt genom att ge x, y och z värden till 0. Materialet beslöt jag att tas in på en gång och är bäddade in så fort applikationen körs. Därefter skapar man en sfär och texturerar den med panoramabilden. För att visa bilden rätt bör den vara spegelvänd. Då man texturerar sfären på utsidan men tittar på bilden från sfärens mitt ser panoramabilden spegelvänd ut, ifall man inte har svängt bilden horisontalt. Till sist gäller det att renderera scenen, kameran och viewporten samt lägga in lyssnare, som möjliggör interaktiviteten.

5.1.3 Interaktivitet

Lyssnarna kollar ifall musen eller någon av meny objekten klickas och bestämmer vilken funktions som körs.

```
addEventListener(Event.ENTER_FRAME, onEnterFrame);
viewport.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
viewport.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
viewport.addEventListener(MouseEvent.ROLL_OVER, onMouseUp);
```

Grundfunktionen som körs automatiskt är *onEnterFrame()*. Den körs konstant och har som huvuduppgift att rendera applikationen. I samma funktion har jag också byggt upp rotationen av kameran. Det är naturligt att ha rotationen i samma funktion, för då man svänger på kameran är applikationen tvungen att renderera den nya bilden.

Rotationen är sammankopplad var musen befinner sig på scenen. Ju längre bort från centrum man är, desto snabbare roterar kameran. Rotationen fungerar i alla riktningar med samma hastighet. Roterar man kameran rakt upp eller motsvarande rakt ner, finns en förhindrande funktion som undviker kameran att gå över. Ifall denna funktion inte skulle finnas, riskerar man att kameran hamnar upp och ner. Nedan är funktionen för *onEnterFrame()*.

```
private function onEnterFrame(param1:Event = null) : void
{
	if (myMouseDown)
	{
		camera.rotationY += (mouseX - stage.width / 2) / 90;
		camera.rotationX += (mouseY - stage.height / 2) / 90;
		if (camera.rotationX <= -90)
		{
```

```
                camera.rotationX = -90;
            }
            else if (camera.rotationX >= 90)
            {
                camera.rotationX = 90;
            }

            var rotate:Number = camera.rotationY - (stage.width / 2 -
            mouseX) / 20;
            TweenLite.to(camera, 0.75,{ rotationY: rotate,
            ease:Cubic.easeOut});
        }
        renderer.renderScene(scene, camera, viewport);
}
```

Rotationen fungerar även med piltangenterna på tangentbordet. Rotationen bygger på samma princip som i exemplet ovan.

Förutom rotationen har jag ett litet externt animeringsverktyg: TweenLight. Genom at använda detta verktyg, får man rotationen att röra sig smidigt och resultatet blir naturligt för användaren.

Menyn är byggd på olika bilder som placeras i viss ordning. De två första är huvudmenyer och har undermenyer medan de två sista är direkta länkar till respektive rummet. Figur 18 visualiserar menyn där huvudmenyn *Auditorium* är öppnad. Öppnandet och stängandet av huvudmenyn är animerad och sker med hjälp av TweenLight animeringsverktyget. Animeringen sker beroende ifall någon av huvudmenyerna är redan öppnad. Till exempel ifall *Seminarie* menyn är öppen och man vill öppna *Auditorium* stängs först *Seminarie* innan *Auditorium* öppnas med en liten dröjning.

```
public function auditoriumDropDown(event:MouseEvent):void
{
        if (audiDrop)
        {
            if ((!seminarieDrop) && (audiDrop))
            {
                seminarieEaseUp();
                var timer:Timer = new Timer(500, 1);
                timer.addEventListener(TimerEvent.TIMER, delay);
                timer.start();
                function delay(e:TimerEvent):void
                {
                    auditoriumEaseDown();
                }
            }
```

```
            else
            {
                  seminarieEaseUp();
                  auditoriumEaseDown();
            }
      }
      else
      {
            auditoriumEaseUp();
      }
}
```

Funktionerna *EaseUp()* och *EaseDown()* bestämmer animeringen.

I presentationen är det även möjligt att zooma via zoomknapparna. För att undvika oändlig zoom har jag byggt ut vissa begränsningar. Nedan ett exempel på hur zoom in fungerar. Övergången mellan stegen sker dynamiskt och är animerade med TweenLite verktyget. Zoom funktionen fungerar även med musens hjul.

```
public function zoomInClick(event:MouseEvent):void
{
      zoomNumber = camera.zoom + (camera.zoom / 2) / 2;
      TweenLite.to(camera, 0.25,{ zoom: zoomNumber,
ease:Linear.easeNone});
      if (camera.zoom > 1.7)
      {
            zoomin.enabled = false;
            zoomin.alpha = 0.5;
      }
      else if (camera.zoom < 1.35)
      {
            zoomout.enabled = true;
            zoomout.alpha = 1;
      }
}
```

Därtill har jag en kontrollerare *activePanorama()* funktion som går igenom vilken panorama som är aktiv och inbäddar vissa element i presentationen. Dessa är bland annat informationsrutan som ger kort översikt över rummet ifråga. Dessutom begränsar funktionen vilket rum som är aktivt i menyn och hindrar att öppna samma rum som redan är aktiv. Nedan ett utdrag ur funktionen.

```
if (panoChooser == "stora_auditoriet")
      {
            stora_torget.enabled = true;
            lilla_auditoriet.enabled = true;
            stora_auditoriet.enabled = false;
```

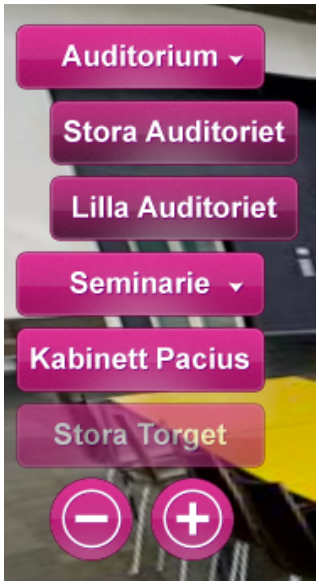
```
        zoomin.alpha = 1;
        zoomout.alpha = 1;
        zoomin.enabled = true;
        zoomout.enabled = true;

        stora_torget.alpha = 1;
        lilla_auditoriet.alpha = 1;
        stora_auditoriet.alpha = 0.6;

        show_info.alpha = 0;
        stora_a_info.alpha = 1;
        lilla_a_info.alpha = 0;
        stora_torget_info.alpha = 0;
        d4106_info.alpha = 0;
        b3320_info.alpha = 0;
        pacius_info.alpha = 0;
}
```

5.1.4 Grafik

De grafiska elementen representerar menyn samt en liten infobox. Uppdragsgivaren önskade att färgtemat skulle följa Arcadas nya färger, svart, vit och pink. I Figur 18 kan man se huvud- samt undermeny för Auditorium. Zoom funktionen körs via plus och minus figurerna.



Figur 18. Meny.

Varje element har en egen lyssnare och egna funktioner. Då man vill navigera till ett nytt rum klickar man på menyn och då körs funktionen för den specifika knappen. För

att skapa taxonomi grundades två huvudmenyer och två direkta spår till byte av panoraman. Vid klick av huvudmenyn skjuts menyknapparna under nedåt medan undermenyn glider fram. Funktionen baserar sig enkelt på att förflytta menybilderna nedåt. Funktionen kan realiseras i exempelkoden nedan.

```
private function auditoriumEaseDown():void
{
	TweenLite.to(seminarie, 0.6,{y: 235, ease:Cubic.easeIn});
	TweenLite.to(pacius, 0.6, {y: 280, ease:Cubic.easeIn});
	TweenLite.to(stora_torget, 0.6, {y: 325, ease:Cubic.easeIn});
	TweenLite.to(stora_auditoriet, 0.6,{y: 145, alpha: 1,
ease:Cubic.easeIn});
	TweenLite.to(lilla_auditoriet, 0.6,{y: 190, alpha: 1,
ease:Cubic.easeIn});
	TweenLite.to(zoomin, 0.6, {y: 370, ease:Cubic.easeIn});
	TweenLite.to(zoomout, 0.6, {y: 370, ease:Cubic.easeIn});
	audiDrop = false;
	if (panoChooser == "stora_auditoriet")
	{
		TweenLite.to(stora_auditoriet, 0.6,{y: 145, alpha: 0.6,
ease:Cubic.easeIn});
		TweenLite.to(lilla_auditoriet, 0.6,{y: 190, alpha: 1,
ease:Cubic.easeIn});
	}
	else if (panoChooser == "lilla_auditoriet")
	{
		TweenLite.to(stora_auditoriet, 0.6,{y: 145, alpha: 1,
ease:Cubic.easeIn});
		TweenLite.to(lilla_auditoriet, 0.6,{y: 190, alpha: 0.6,
ease:Cubic.easeIn});
	}
}
```

Vid val av nytt panorama förstör man i praktiken först det nuvarande panoramant, väljer hurdan textur man skall visa d.v.s. vilket rum och sedan skapar en ny sfär med den valda texturen. Funktionen nedan visar byte av panorama. Samtidigt återställs kamerans zoom och startpunkt. Informationsfönstret visas automatiskt vid varje byte av utrymme.

```
public function stora_auditoriet_click(event:MouseEvent):void
{
	panoSphere.material.bitmap.dispose();
	panoSphere.material.destroy();
	scene.removeChild(panoSphere);
	panoChooser = "stora_auditoriet";
	generateScene();
```

```
        camera.zoom = 1.5;
        camera.rotationX = 2;
        camera.rotationY = 11;
        stora_a_info.x = (stage.width / 2) - (stora_a_info.width / 2);
        stora_a_info.y = stage.height - stora_a_info.height;
        activePanorama();
}
```

5.2 Fotografering

Detta kapitel redogör för de centrala delarna av fotograferingsarbetet. Examensarbetet handlar inte om hurdan utrustning som är optimal för panoramafotograferandet, varför jag endast presenterar vad jag själv använt för utrustning.

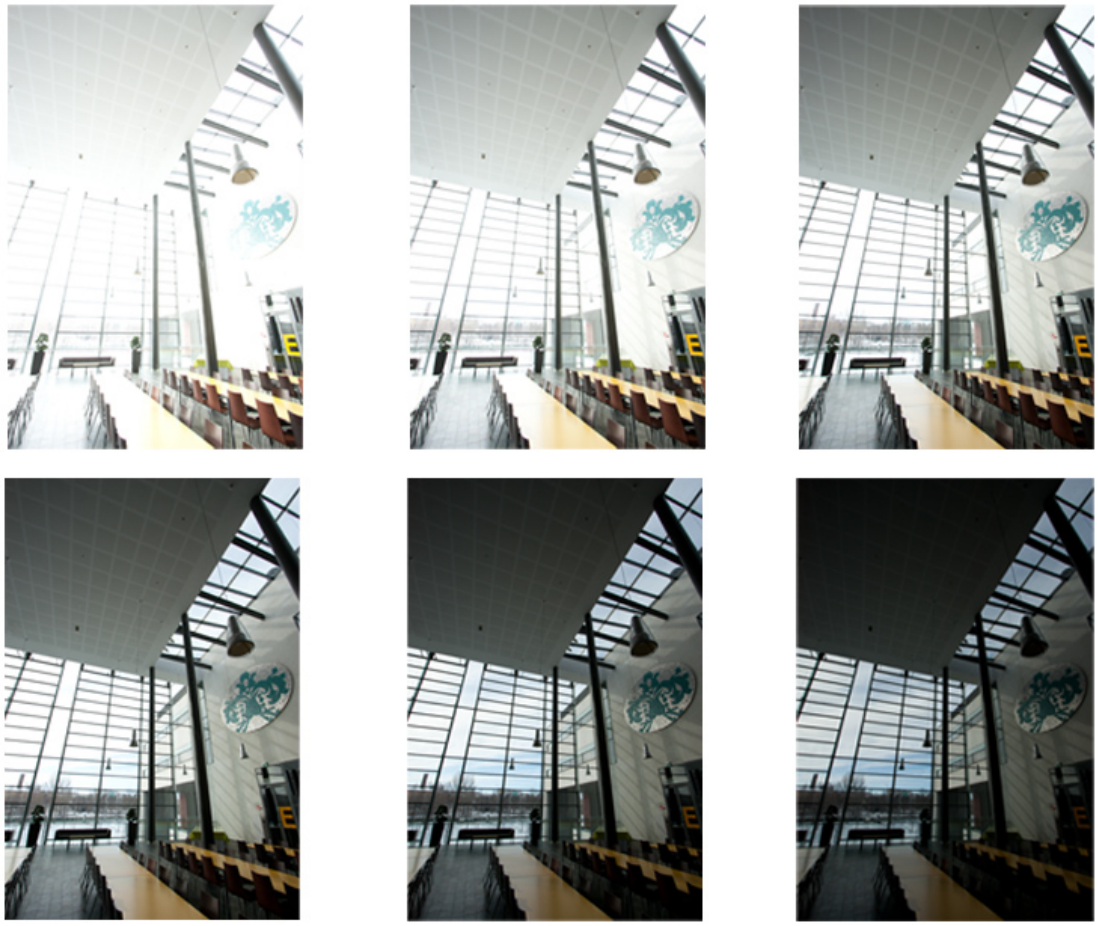
5.2.1 Utrustning och arbetsmetoder

Det finns mängder av olika sätt att fotografera panoramabilder. I detta examensarbete användes Canon EOS 450D kamerahus och Canon EF-S 10-22/3.5-4.5 USM vidvinkel lins. Dessutom användes Manfrotto stativ och Nodal Ninja 3 panoramahuvud tillsammans med en utjämnare. Figur 19 visar hur uppsättningen ser ut.

Figur 19. Bild på kamera uppsättning med panoramahuvud (Panophoto).

Fotografierna ät tagna med HDR (High Dynamic Range) tekniken. Detta innebär att man tar tre eller flera exponeringar i samma riktning och sedan sammansmälter dem. På så sätt får man fram material både från mörka och ljusa områden. (HDR 2011) Kameran

som användes i detta examensarbete har en funktion som automatiskt tar en normal, en över- samt en underexponerad bild. Figur 20 visar ett exempel på 6 exponeringar.



Figur 20. Exempel på 6 exponeringar.

Bilderna är fotograferade i RAW format. Detta betyder att de inte är bearbetade på något sätt. RAW bilder är även kallade för digitala negativ. Bilderna justeras i Adobe Camera Raw där bl.a. vitbalansen och färgtonerna rättas. Sedan sparas bilderna till "positiv" filformat, oftast TIFF.

Då de enskilda bilderna är justerade och sparade till rätt format påbörjas ihop montering till ett panorama. Detta sker med hjälp av programmet PTgui. Mera om detta skede i kapitel 5.2.3.

Slutligen konverteras panorama bilden till swf format. Detta sker genom att importera bilden till Adobe Flash och spara projektet. Då skapas en bild i formatet swf.

5.2.2 Fotografering

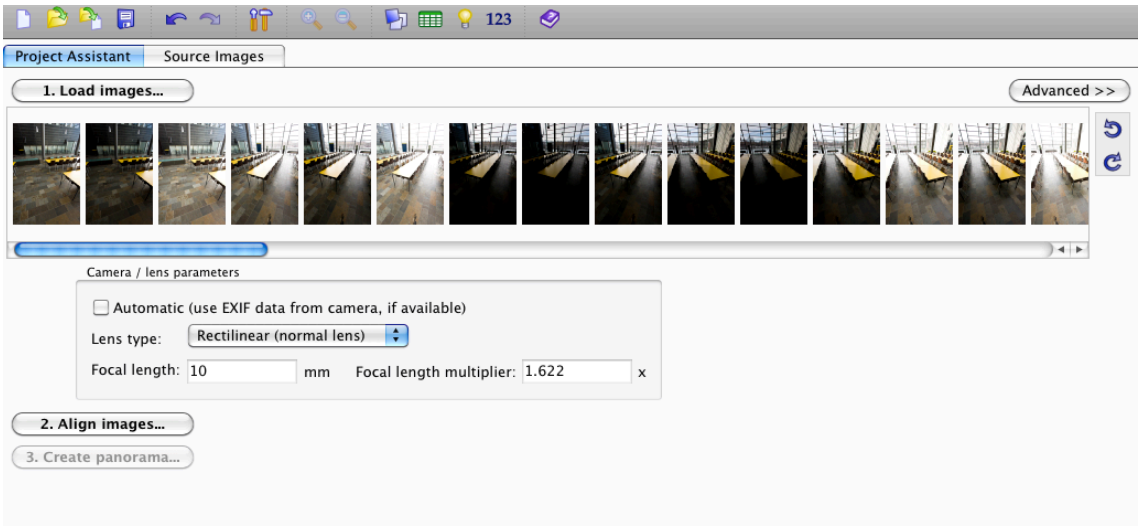
Eftersom jag fotograferade med en vidvinkellins vid 10 mm, var jag tvungen att fotografera panoraman i två plan. Detta betyder att jag hade kameran i 30° och 120° vinkel då 0° pekar rakt nedåt. För att få en full 360° vy måste jag alltså fotografera 8 bilder med 45° mellanrum. Då fotografierna togs med HDR tekniken tredubblas antalet fotografier. Tillsammans blev det minst 48 bilder. Då vissa exceptionellt ljusa områden måste fotograferas med flera än tre exponeringar ökade antalet enskilda fotografier upp till 72 eller flera. Då kameran är monterad på ett stativ och andra inställningar klara, gäller det endast att trycka på avlösaren och svänga kameran an efter tre bilder är tagna från denna vinkel.

Presentationen visar en full 360° vy i alla riktningar. Detta innebär att också zenit och nadir är fotograferade. Zenit är det man anser är rakt uppåt och nadir motsvarande rakt nedåt. (Zenit 2011) Att fotografera zenit är problemfritt då man har kameran på ett stativ, men nadir orsakar mera problem eftersom man vill ju inte att stativet syns på bilden. Då gäller det att montera om stativet så det går att fotografera nedåt från maximal höjd. Är man utomhus, eller någonstans där det är mycket ljus har man inget problem med stativet för man kan fotografera nadir bilderna på fri hand.

Då det gäller att fotografera panoramabilder är kamerans inställningar en av de viktigaste aspekterna man skall fundera på. För att bilden skall ha liknande vitbalans och exponering bör kameran vara inställd på manuellt läge. Bländaren och slutartiden skall vara lika oberoende åt vilket håll man fotograferar. För att få skärpedjupet skarpt både nära och längre bort på bilden torde man ställa in bländaren på ett värde nära 10. Fokus skall vara manuellt på oändligt läge. Då blir bilden lika skarp i förgrunden som i bakgrunden.

5.2.3 Ihopmontering

Ihopmontering som även kallas för stichning är den mest betydande arbetsdelen efter själva fotograferandet. Med hjälp av programmet PTgui (Panorama Tools graphical user interface) är själva processen behändig att följa. Man laddar in alla bilder i programmet, förutom zenit och nadir. Viktigt är att de är i rätt ordning, bredvid varande vinklar skall följas av varandra.



Figur 21. Exempel på programmet PTgui.

Programmet skapar kontrollpunkter som förenar bilderna automatiskt. Man kan vid behov manuellt justera och mata in flera kontrollpunkter ifall resultatet verkar dåligt eller fel. Har man fotograferat i HDR och har lika många exponeringar på alla bilder uppfattar PTgui detta och arbetar samma bilder med lika exponering ihop utan att blanda in de andra exponeringarna. Dessutom blandar programmet de olika exponeringarna till ett grundpanorama. Slutliga filen består av fullständiga panoraman i olika lager av respektive exponering. Detta är till stor nytta då man har flera exponeringar, men vill bara använda en del av dem. I detta examensarbete stichades bilderna till ett sfäriskt panorama. Vid behov kan man även montera olika former av panoraman med programmet, såsom kub eller cylinder.

Slutligen ger man in värden på panoramats storlek och i vilket format de skall sparas. Sedan gäller det att redigera den färdiga panorama i Adobe Photoshop.

5.2.4 Redigering

Resultatet av en HDR-panorama som PTgui har stichat ihop är en photoshop fil där alla lika exponeringar ligger på olika lager. Fördelar med detta är att man kan med hjälp av maskering bestämma hur mycket av respektive lager som blir synligt. I detta examensarbete fotograferades rum med stora fönster. Då blir ljus skillnaden mellan utomhus och inomhus stor. Maskerar man bort av ljusa exponeringar och maskerar fram av mörka

maskeringen skapar man synlighet ut, utan att förlora ljuset inomhus. Exempel på detta är visualiserat i Figur 22.

Figur 22. Jämförelse mellan vanlig- och HDR-fotografi.

Även om PTgui räknar och blandar ihop bilderna mycket bra uppstår det dock små fel. Dessa fel är oftast linjer som har ett hack. Vid redigeringsskedet elimineras dessa fel genom att justera närområdet av felet. Andra fel som uppstår kan vara dammfläckar som uppstått av smuts på linsen. För att bilden skall se felfri ut, elimineras dessa med hjälp av *clone* eller *patch* verktygen.

För att sammanfoga zenith och nadir bilderna justeras panorama-bilden med PTgui så att bilden konverteras till en vinkel nedåt och motsvarande en bild uppåt. Då man har en bild med en polygon i mitten kan man lägga fotografin som är tagen i samma riktning och smälta in den över polygonen så att bilden ser riktig ut (Figur 23). Därefter konverteras bilden tillbaka till normal-panorama med PTgui.

Figur 23. Exempel på nadir före och efter redigering.

Innan den slutliga bilden är färdig skall färgnyanser och kontraster justeras så de passar omgivningen. För att spara bästa möjliga kvalitet sparas bilden slutligen i TIFF format.

5.3 Uppbyggnad

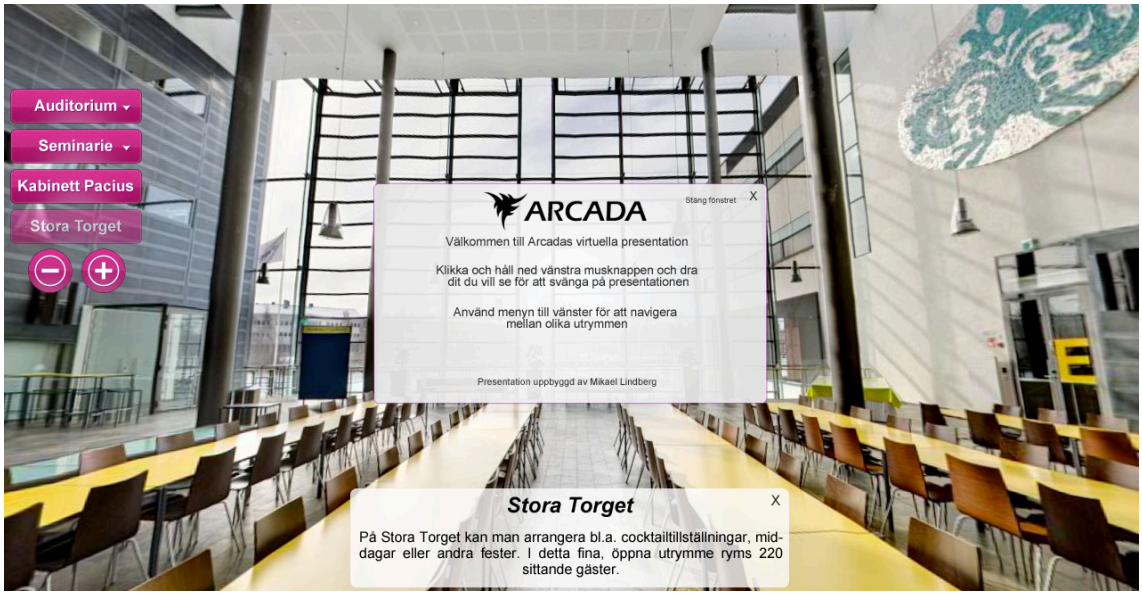
Konvertering från tiff-format till swf-format sker på följande sätt: den färdiga panorama i TIFF format importeras in till Adobe Flash, bredd och höjd justeras till 2826 respektive 1413 pixel. Sedan publiceras Flash filen till swf-format. Detta upprepas för alla panoraman. Detta arbetsskede är nödvändig, då man vill spara på filstorlek. En panoramabild i tiff-format är ca 12 MB medan motsvarande bild i swf-format ligger på 500 kB. Bildkvaliteten försämras en aning vid komprimering till swf-format.

Då alla swf-filer är klara samlar man dem till samma ställe på datorn och definierar filstigen så att Papervision3D hittar dem. På samma ställe skall man även samla alla grafiska element som används i presentationen.

Slutligen när alla delmoment är klarade kan man bygga panoramapresentationen genom att använda Papervision3D motor. Panoramapresentationen skalas automatiskt till den upplösning som användaren har till förfogandet. Detta är en standard funktion i Papervision3D.

6 DISKUSSION

Den slutliga produkten (Figur 24) blev en fullt fungerande presentation över sex olika utrymmen i Arcada. Färgtemat i presentationen baserar sig på färger från Arcadas webbplats. Navigeringen fungerade bra och övergången mellan de olika utrymmena gick smidigt.



Figur 24. Panoramapresentationens framsida.

6.1 Papervision3D

Ett mål med detta examensarbete var att identifiera ifall Papervision3D motor passar för att bygga panoramapresentationer. Papervision3D medför en hel del funktionalitet och effekter. Att ta vara på alla dessa kan vara svårt, speciellt ifall man inte har tidigare erfarenhet av liknande system. Utan denna kunskap kan projektet fortlöpa långsamt och en del problem kunde lösas på ett enklare sätt än hur de eventuellt löstes. Grundfunktionerna är dock möjliga att behärska utan större fördjupning och byggandet av panoramapresentation lyckades bra. Kapitel 3.2 beskriver noggrannare huvudklasserna som utgör grundfunktionerna.

6.1.1 Fördelar

Trots den stora mängden effekter och funktionalitet är det dock möjligt att genom dokumentationen behärska funktionerna. Dokumentationen består av alla klasser och förklarar deras byggnad och användning på ett mycket effektivt sätt.

En annan stor fördel är Papervision3D:s popularitet. På internet kan man hitta otaliga exempel på olika Papervision3D projekt. Webbsidan Lynda.com presenterar några mycket bra exempelvideon av olika Papervision3D projekt. Detta gör det enkelt att hitta lösningar till problem. Förutom exempel kan man hitta flera olika forum där man kan få hjälp av andra utvecklare. Papervision3D har också en stor samhörighetsgrupp. Denna grupp är villig att hjälpa vid problem. Det finns också ett antal böcker skriva för att utveckla Papervision3D projekt, där man kan leta efter problemlösningar.

6.1.2 Nackdelar

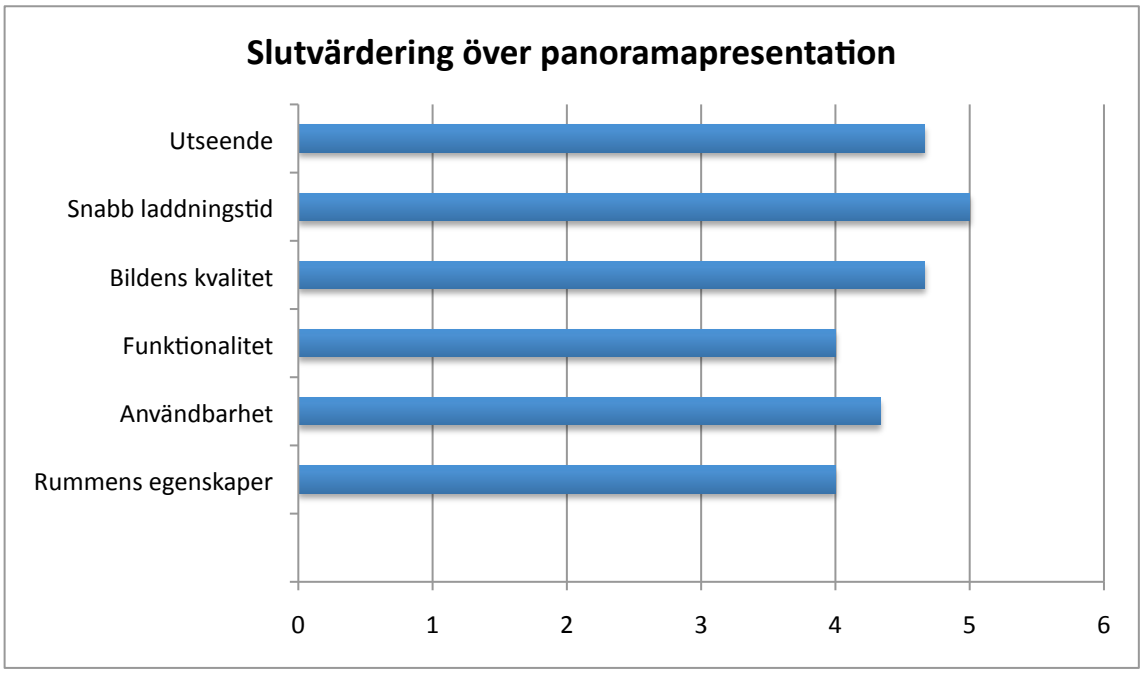
Stora nackdelen med Papervision3D är att den officiella utvecklingen har slutat. Papervision3D är skriven med öppen källkod, vilket leder till att vem som helst kan fortsätta utvecklandet. Det är dock ganska osannolikt, eftersom det skulle kräva att källkoden skrivs om från början, för att kunna dra nytta av nyaste Flash Player versionens egenskaper.

Annan nackdel är att Papervision3D inte är optimerad för nyaste Flash Player, vilken stöder hårdvara accelerering. Detta gör att andra motsvarande system tar över i popularitet. I kapitel 2.4 har jag listat några av de närmaste konkurrenterna till Papervision3D. Av dem ser jag Away3D och Alternativa3D som starkaste fortsättarna.

Papervision3D har små luckor, även om den senaste versionen är stabil och fungerande. Ett litet problem som kom fram under utvecklandet av presentationen var ett minnesproblem. Applikationens minne läckte, utan att man kunde stoppa det. I praktiken betyder det att om man har programmet öppen en längre stund blir den långsam och trög. Läckaget var dock så liten att det inte borde orsaka problem i mitt projekt.

6.2 Presentationen

Presentationen testades genom eye tracker för att granska ifall de praktiska fungerar mellan användaren och programmet. Samma utvärdering som i intervjun gjordes efter testet, skalan ligger på 0-5. Kapitel 4.2 redogör hur eye tracker testet genomfördes. Resultatet (Figur 25) bevisar att produkten fungerar och användarna hade goda erfarenheter, speciellt lyckades jag skapa en laddningstid som alla var ense om.



Figur 25. Slutvärdering över panoramapresentation.

Ett av syften var att höja intresset på uthyrningen av utrymmen i Arcada. Svaret till detta kan konstateras först efter att presentationen varit i bruk en tid och en uppföljning har gjorts.

Med hjälp av feedback av testpersonerna kunde presentationen optimeras för att passa användarens behov och några små tilläggs funktioner byggdes ut på begäran.

Personligen är jag av den åsikten att projektet lyckades och hoppas på att presentationen används i framtiden. Med mycket liten Papervision3D bakgrund och liten erfarenhet av ActionScript programmering har jag lärt mig mycket under projektets gång. Lösandet av problem var ibland mycket svårt, men med hjälp av exempel och böcker lyckades jag orientera mig rätt för att lösa problemen.

6.3 Slutsats

Jag valde Papervision3D för detta projekt eftersom jag redan i ett tidigare skede hade bekantat mig med motorn. Det kändes naturligt att fortsätta jobba med motorn. Utvecklandet av Papervision3D hade slutat fick jag reda på under arbetet men gjorde beslutet att fortsätta eftersom Papervision3D fortfarande används och det finns mycket material som man kan dra nytta av. Eftersom panoramapresentation inte kräver extra effekter fungerar Papervision3D utmärkt. Kunde detta ha gjorts med annan 3D motor? Ja, det är möjligt att göra panoramapresentationer med nästan vilken 3D motor som helst. Det skulle dock kräva en fördjupning i ett nytt verktyg.

Att programmera med ett avslutat projekt medföljer alltid risker. Stöter man på ett problem som ligger i kärnkoden har man dåliga chanser att lösa den. Då andra projekt och internet överlag utvecklas konstant kommer Papervision3D inte att hänga med, eftersom programmet blir föråldrat.

KÄLLOR

Allen, Chris; Arnold, Wade; Balkan, Aral; Cannasse, Nicolas; Grden, John; Gunesch, Moses; Hughes, Marc; MacDonald, R. Jon; Zupko, Andy. 2008, *The Essential Guide to Open Source Flash Development,* Friends of ED, 382 s.

Lively, Michael. 2010, *Professional Papervision3D*, Storbritannien: John Wiley & Sons, Limited, 721s.

Winder, Jeff; Tondeur, Paul. 2009, *Papervision3D Essentials,* Packt Publishing, 407 s.

MIT License. 2011, Wikipedia, publicerad 19.2.2011, Tillgänglig: http://en.wikipedia.org/wiki/MIT_License Hämtad 24.2.2011.

Kallonen, Matti. 2010, *Papervision3D:n käyttö verkkosivuilla,* Examensarbete, Riihimäki: Hämeen Ammattikorkeakoulu, Mediatekniikka.

Ulloa, Carlos. 2009, Papervision is Shifting Gears, Publicerad 13.10.2009, Tillgänglig: http://blog.papervision3d.org/2009/10/13/papervision3d-is-shifting-gears/ Hämtad 24.2.2011.

Grden, John. 2007, Papervision3D to merge Away3D features, Publicerad 16.5.2007, Tillgänglig: http://blog.papervision3d.org/2007/05/16/papervision3d-to-merge-away3d-features/ Hämtad 24.2.2011.

Zadorozhny, Alexander. 2007, Away3D Team, Publicerad 16.6.2007, Tillgänglig: http://away3d.com/team Hämtad 25.2.2011.

Sandy. 2009, Tillgänglig: http://www.flashsandy.org/ Hämtad 25.2.2011.

Sophie. 2009, Tillgänglig: http://www.sophie3d.com/ Hämtad 25.2.2011.

Wavefront .obj. 2011, Wikipedia, publicerad 28.1.2011, Tillgänglig: http://en.wikipedia.org/wiki/Wavefront_.obj_file Hämtad 25.2.2011.

Alternativa. 2010, Tillgänglig: http://alternativaplatform.com/en/ Hämtad 25.2.2011.

PapervisionX. 2009, PapervisionX – what it is and what it isn´t, Publicerad 16.3.2009, Tillgänglig: http://blog.papervision3d.org/2009/03/16/papervisionx-what-it-is-and-what-it-isnt/ Hämtad 1.3.2011.

Papervision. 2010a, Tillgänglig: http://blog.papervision3d.org Hämtad 1.3.2011.

HDR. 2011, Wikipedia, publicerad 27.2.2011, Tillgänglig: http://en.wikipedia.org/wiki/High-dynamic-range_imaging Hämtad 3.3.2011.

Zenit Nadir. 2011, Wikipedia, pulicerad 17.2.2011, Tillgänglig: http://sv.wikipedia.org/wiki/Zenit Hämtad 4.3.2011.

Zupko, Andy. 2008a, FogFilter, Publicerad 2.6.2008. Tillgänglig: http://blog.zupko.info/?p=134 Hämtad 13.3.2011.

Zupko, Andy. 2008b, ShadowCaster, Publicerad 10.7.2008. Tillgänglig: http://blog.zupko.info/?p=146 Hämtad 19.3.2011.

Plane. 2011, Wikipedia, publicerad 10.3.2011, Tillgänglig: http://simple.wikipedia.org/wiki/Plane_(mathematics) Hämtad 15.3.2011.

Sfär. 2011, Wikipedia, publicerad 16.2.2011, Tillgänglig: http://sv.wikipedia.org/wiki/Sfär Hämtad 15.3.2011.

Cube. 2011, Wikipedia, publicerad 26.3.2011, Tillgänglig: http://en.wikipedia.org/wiki/Cube Hämtad 1.4.2011.

Papervision. 2010b, Papervision API, Tillgänglig: http://papervision3d.googlecode.com/svn/trunk/as3/trunk/docs/index.html Hämtad 15.3.2011.

Papervision. 2010c, Papervision3D Google Code, Tillgänglig: http://code.google.com/p/papervision3d/ Hämtad 17.3.2011.

Tessellation. 2011, Wikipeia, publicerad 20.2.2011, Tillgänglig: http://sv.wikipedia.org/wiki/Tessellation Hämtad 17.3.2011.

Tobii T120. 2010, Tobii Eye Tracker. Tillgänglig: http://www.tobii.com/en/analysis-and-research/global/products/hardware/tobii-t60t120-eye-tracker/ Hämtad 31.3.2011

Panophoto. Tillgänglig: http://homepage.mac.com/gibie76/panophoto/00/Materiel/Tete_pano/Nodal_Ninja/NN3/NN3-2.jpg Hämtad 12.4.2011

BILAGOR

Bilaga 1. Intervjufrågor

Bilaga 2. Eye Tracking forskningsfrågor

Bilaga 3. Fullständig källkod över panoramapresentationen

Bilaga 4. Resultat över eye tracking testet (DVD)

Intervju om virtuell presentation för Arcada

Plats: **Specialisering:**

Datum:

1. **Vilken uppfattning har du om virtuell panorama presentation?**
   Hur bekant är du med ämnet?

2. **Ifall du inte är alls bekant med ämnet, vad har du för uppfattning om virtuell panorama presentation?**

3. **Vilka brister har du funnit då man skall söka information om Arcadas utrymmen eller allmänt om utrymmen man vill hyra?**

4. **Hurdan information anser du att är bra och viktig och vad som är nödvändigt?**

5. **Tycker du att navigeringen skall ske via bottenplan eller meny?**

6. **Bedöm hur viktigt följande saker är i en virtuell presentation.**
   (där 1 motsvarar inte alls nödvändigt och 5 mycket viktigt)

Utseende	1 2 3 4 5
Snabb laddningstid	1 2 3 4 5
Bildens kvalitet	1 2 3 4 5
Funktionalitet	1 2 3 4 5
Användbarhet	1 2 3 4 5
Rummens egenskaper	1 2 3 4 5
Smarttags/hotspots i panoraman	1 2 3 4 5

7. **Är det något du vill tillägga?**

ARCADA
NYLANDS SVENSKA YRKESHÖGSKOLA

Forskningsfrågor för Eye Tracker

1. **Läser användaren instruktionerna?**

1.1 **Hur lång tid vistas innan instruktionen göms?**

2. **Vad gör användaren efter instruktionerna?**

3. **Förstår användaren menyn och dess knappar?**

3.1 **Används zoom?**

4. **Vilka områden centreras blicken mest?**

4.1 **Vilka blir utanför?**

5. **Tittar användaren i taket och/eller golvet?**

6. **Fungerar användargränssnittet enligt "Jakob Nielsens" teori om användarvänlighet på webben?**

Fullständig källkod over panoramapresentationen

```
<?xml version="1.0" encoding="utf-8"?>
<mx:Application xmlns:mx="http://www.adobe.com/2006/mxml" layout="absolute" frameRate="30"
applicationComplete="init()">
            <!-- creationComplete="init()"-->
            <mx:Script>
                        <![CDATA[
                                    include "Panorama.as";
                        ]]>
            </mx:Script>

            <mx:UIComponent x="0" y="0" width="40" height="41" id="PV3D">
            </mx:UIComponent>
            <mx:UIComponent id="statsview">
            </mx:UIComponent>

            <mx:Image x="30" y="145" id="d4106" source="assets/menu/D4106.png" alpha="0" />
            <mx:Image x="30" y="145" id="b3320" source="assets/menu/B3320.png" alpha="0" />
            <mx:Image x="10" y="145" id="seminarie" source="assets/menu/seminarie.png"
visible="true" />
            <mx:Image x="30" y="100" id="lilla_auditoriet"
source="assets/menu/lilla_auditoriet.png" alpha="0" />
            <mx:Image x="30" y="100" id="stora_auditoriet"
source="assets/menu/stora_auditoriet.png" alpha="0" />
            <mx:Image x="10" y="100" id="auditorium" source="assets/menu/auditorium.png"
visible="true"/>

            <mx:Image x="10" y="190" id="pacius" source="assets/menu/pacius.png" />
            <mx:Image x="10" y="235" id="stora_torget" source="assets/menu/stora_torget.png"
/>

            <mx:Image x="90" y="280" id="zoomin" source="assets/menu/zoomin.png" />
            <mx:Image x="30" y="280" id="zoomout" source="assets/menu/zoomout.png" />
            <mx:Image x="-500" id="stora_a_info" source="assets/stora_auditoriet_info.png"
alpha="0" />
            <mx:Image x="-500" id="lilla_a_info" source="assets/lilla_auditoriet_info.png"
alpha="0" />
            <mx:Image x="-500" id="stora_torget_info" source="assets/stora_torget_info.png"
alpha="0" />
            <mx:Image x="-500" id="d4106_info" source="assets/d4106_info.png" alpha="0" />
            <mx:Image x="-500" id="b3320_info" source="assets/b3320_info.png" alpha="0" />
            <mx:Image x="-500" id="pacius_info" source="assets/pacius_info.png" alpha="0" />
            <mx:Image x="-100" id="show_info" source="assets/menu/info.png" alpha="0" />
            <mx:Image id="popupWindow" source="assets/popup.png" />
</mx:Application>
```

Panorama.as

```
import com.greensock.*;
import com.greensock.easing.*;

import flash.display.*;
import flash.events.*;
import flash.net.*;
import flash.ui.ContextMenu;
import flash.ui.ContextMenuBuiltInItems;
import flash.ui.ContextMenuItem;

import mx.containers.*;
import mx.controls.*;
import mx.core.*;
import mx.events.*;
import mx.styles.*;
import mx.utils.object_proxy;

import org.papervision3d.cameras.*;
import org.papervision3d.core.ns.pv3dview;
import org.papervision3d.events.*;
import org.papervision3d.materials.*;
import org.papervision3d.objects.primitives.*;
import org.papervision3d.render.*;
import org.papervision3d.scenes.*;
import org.papervision3d.view.*;
import org.papervision3d.view.stats.StatsView;


private var renderer:BasicRenderEngine;
private var scene:Scene3D;
private var camera:Camera3D;
private var viewport:Viewport3D;
private var panoSphere:Sphere;
private var sphereMat:MovieMaterial;
private var myMouseDown:Boolean = false;
private var audiDrop:Boolean = true;
private var seminarieDrop:Boolean = true;
private var panoChooser:String;
private var stats:StatsView;
private var myMenuItem:ContextMenuItem;
private var contextMenuCustomItems:Array;
public var zoomNumber:Number;
public var keyRotate:Number;

// Panoramas to be shown

[Embed(source="assets/stora_auditoriet.swf")]
private var storaAuditorietAsset:Class;

[Embed(source="assets/lilla_auditoriet.swf")]
private var lillaAuditorietAsset:Class;
```

```
[Embed(source="assets/D4106.swf")]
private var D4106Asset:Class;

[Embed(source="assets/B3320.swf")]
private var B3320Asset:Class;

[Embed(source="assets/pacius.swf")]
private var paciusAsset:Class;

[Embed(source="assets/stora_torget.swf")]
private var storaTorgetAsset:Class;


// Init function, autorun
public function init():void
{

            // Papervision Variables
            viewport = new Viewport3D(800, 600, true, true);
            //pv3Canvas.rawChildren.addChild(viewport); mx:canvas
            PV3D.addChild(viewport);
            scene = new Scene3D();
            camera = new Camera3D();
            renderer = new BasicRenderEngine();

            // Stats View
            /*
            var stats:StatsView = new StatsView(renderer);
            statsview.addChild(stats);*/

            // Setup camera
            positionCamera();

            // Info window
            popup();

            // Setup Materials
            generateScene();

            // Custom right-click options
            rightClick();

            //  ---  Listeners  ---   //
            this.stage.addEventListener(KeyboardEvent.KEY_DOWN, onKeyPress);
            addEventListener(Event.ENTER_FRAME, onEnterFrame);
            addEventListener(MouseEvent.MOUSE_WHEEL, onMouseWheel);
            viewport.addEventListener(MouseEvent.MOUSE_DOWN, onMouseDown);
            viewport.addEventListener(MouseEvent.MOUSE_UP, onMouseUp);
            viewport.addEventListener(MouseEvent.ROLL_OVER, onMouseUp);

            auditorium.addEventListener(MouseEvent.CLICK, auditoriumDropDown);
            seminarie.addEventListener(MouseEvent.CLICK, seminarieDropDown);
```

```
            stora_auditoriet.addEventListener(MouseEvent.CLICK, stora_auditoriet_click);
            lilla_auditoriet.addEventListener(MouseEvent.CLICK, lilla_auditoriet_click);
            d4106.addEventListener(MouseEvent.CLICK, d4106_click);
            b3320.addEventListener(MouseEvent.CLICK, b3320_click);
            pacius.addEventListener(MouseEvent.CLICK, pacius_click);
            stora_torget.addEventListener(MouseEvent.CLICK, stora_torget_click);

            zoomout.addEventListener(MouseEvent.CLICK, zoomOutClick);
            zoomin.addEventListener(MouseEvent.CLICK, zoomInClick);

            stora_a_info.addEventListener(MouseEvent.CLICK, closeInfo);
            lilla_a_info.addEventListener(MouseEvent.CLICK, closeInfo);
            stora_torget_info.addEventListener(MouseEvent.CLICK, closeInfo);
            d4106_info.addEventListener(MouseEvent.CLICK, closeInfo);
            b3320_info.addEventListener(MouseEvent.CLICK, closeInfo);
            pacius_info.addEventListener(MouseEvent.CLICK, closeInfo);

            show_info.addEventListener(MouseEvent.CLICK, showInfo);

            popupWindow.addEventListener(MouseEvent.CLICK, closePopup);

            stora_torget_info.x = (stage.width / 2) - (stora_torget_info.width / 2);
            stora_torget_info.y = stage.height - stora_torget_info.height;

            myMenuItem.addEventListener(ContextMenuEvent.MENU_ITEM_SELECT, copyright);
}

private function generateScene():void
{
            // Setup Materials
            if(panoChooser == "stora_auditoriet")
            {
                        createMaterial(storaAuditorietAsset);
            }

            else if (panoChooser == "lilla_auditoriet")
            {
                        createMaterial(lillaAuditorietAsset);
            }

            else if (panoChooser == "D4106")
            {
                        createMaterial(D4106Asset);
            }

            else if (panoChooser == "B3320")
            {
                        createMaterial(B3320Asset);
            }

            else if (panoChooser == "pacius")
            {
                        createMaterial(paciusAsset);
            }
```

```
            else if (panoChooser == "stora_torget")
            {
                        createMaterial(storaTorgetAsset);
            }

            /*else
            {
                        createMaterial(storaTorgetAsset);
            }
            */
            // Setup Sphere.
            panoSphere = new Sphere(sphereMat, 25000, 35,35);
            scene.addChild(panoSphere);
}


private function onEnterFrame(param1:Event = null) : void
{
            if (myMouseDown)
            {
                        camera.rotationY += (mouseX - stage.width / 2) / 90;
                        camera.rotationX += (mouseY - stage.height / 2) / 90;
                        if (camera.rotationX <= -90)
                        {
                                    camera.rotationX = -90;
                        }
                        else if (camera.rotationX >= 90)
                        {
                                    camera.rotationX = 90;
                        }

                        var rotate:Number = camera.rotationY - (stage.width / 2 - mouseX) /
20;
                        TweenLite.to(camera, 0.75,{ rotationY: rotate, ease:Cubic.easeOut});
            }
            renderer.renderScene(scene, camera, viewport);
            onZoom();
            trace(camera.zoom);
}

private function positionCamera():void
{
            //Setup camera options
            camera.x = camera.y = camera.z = 0;
            camera.focus = 300;
            camera.zoom = 1.5;
            camera.rotationX = -8;
            camera.rotationY = -65;
}

private function createMaterial(Asset:Class):void
{
            //Create the pano material
```

```
            var movieAsset:MovieClipAsset = new Asset();
            sphereMat = new MovieMaterial(movieAsset, false);
            sphereMat.opposite = true;
            sphereMat.animated = true;
            sphereMat.smooth = true;
            sphereMat.interactive = true;
}

private function onMouseDown(param1:Event = null): void
{
            myMouseDown = true;
            sphereMat.smooth =  false;
}

private function onMouseUp(param1:Event = null): void
{
            myMouseDown = false;
            sphereMat.smooth =  true;
}


/* -- Keyboard Rotation -- */

public function onKeyPress(event:KeyboardEvent): void
{
            switch (event.keyCode)
            {
                        case Keyboard.LEFT:
                                    keyRotate = camera.rotationY - (stage.width / 2 -
camera.rotationY) / 30;
                                    TweenLite.to(camera, 0.6,{rotationY: keyRotate,
ease:Linear.easeNone});
                                    break;
                        case Keyboard.RIGHT:
                                    keyRotate = camera.rotationY + (stage.width / 2 -
camera.rotationY) / 30;
                                    TweenLite.to(camera, 0.6,{rotationY: keyRotate,
ease:Linear.easeNone});
                                    break;

                        case Keyboard.DOWN:
                                    if (camera.rotationX <= -90)
                                    {
                                                camera.rotationX = -90;
                                    }
                                    else if (camera.rotationX >= 90)
                                    {
                                                camera.rotationX = 90;
                                    }
                                    keyRotate = camera.rotationX + (stage.width / 2 -
camera.rotationX) / 60;
                                    TweenLite.to(camera, 0.6,{rotationX: keyRotate,
ease:Linear.easeNone});
                                    break;
```

```
                        case Keyboard.UP:
                                    if (camera.rotationX <= -90)
                                    {
                                                camera.rotationX = -90;
                                    }
                                    else if (camera.rotationX >= 90)
                                    {
                                                camera.rotationX = 90;
                                    }
                                    keyRotate = camera.rotationX - (stage.width / 2 -
camera.rotationX) / 60;
                                    TweenLite.to(camera, 0.6,{rotationX: keyRotate,
ease:Linear.easeNone});
                                    break;

                        default:
                                    break;
            }
}

/*  --- MENU FUNCTIONS ---   */

// -- Auditorium --  //
public function auditoriumDropDown(event:MouseEvent):void
{
            if (audiDrop)
            {
                        if ((!seminarieDrop) && (audiDrop))
                        {
                                    seminarieEaseUp();

                                    var timer:Timer = new Timer(600, 1);
                                    timer.addEventListener(TimerEvent.TIMER, delay);
                                    timer.start();

                                    function delay(e:TimerEvent):void
                                    {
                                                auditoriumEaseDown();
                                    }
                        }
                        else
                        {
                                    seminarieEaseUp();
                                    auditoriumEaseDown();
                        }
            }
            else
            {
                        auditoriumEaseUp();
            }
}
// --  Seminarie -- //
public function seminarieDropDown(event:MouseEvent):void
```

```
{
            if (seminarieDrop)
            {
                        auditoriumEaseUp();
                        seminarieEaseDown();
            }
            else
            {
                        seminarieEaseUp();
            }
}

/* -- Menu easing functions -- */

private function auditoriumEaseDown():void
{
            TweenLite.to(seminarie, 0.6,{y: 235, ease:Cubic.easeIn});
            TweenLite.to(pacius, 0.6, {y: 280, ease:Cubic.easeIn});
            TweenLite.to(stora_torget, 0.6, {y: 325, ease:Cubic.easeIn});
            TweenLite.to(stora_auditoriet, 0.6,{y: 145, alpha: 1, ease:Cubic.easeIn});
            TweenLite.to(lilla_auditoriet, 0.6,{y: 190, alpha: 1, ease:Cubic.easeIn});
            TweenLite.to(zoomin, 0.6, {y: 370, ease:Cubic.easeIn});
            TweenLite.to(zoomout, 0.6, {y: 370, ease:Cubic.easeIn});
            audiDrop = false;
            if (panoChooser == "stora_auditoriet")
            {
                        TweenLite.to(stora_auditoriet, 0.6,{y: 145, alpha: 0.6,
ease:Cubic.easeIn});
                        TweenLite.to(lilla_auditoriet, 0.6,{y: 190, alpha: 1,
ease:Cubic.easeIn});
            }
            else if (panoChooser == "lilla_auditoriet")
            {
                        TweenLite.to(stora_auditoriet, 0.6,{y: 145, alpha: 1,
ease:Cubic.easeIn});
                        TweenLite.to(lilla_auditoriet, 0.6,{y: 190, alpha: 0.6,
ease:Cubic.easeIn});
            }
}

private function auditoriumEaseUp():void
{
            TweenLite.to(seminarie, 0.6,{y: 145, ease:Cubic.easeIn});
            TweenLite.to(pacius, 0.6, {y: 190, ease:Cubic.easeIn});
            TweenLite.to(stora_torget, 0.6, {y: 235, ease:Cubic.easeIn});
            TweenLite.to(stora_auditoriet, 0.6,{y: 100, alpha: 0, ease:Cubic.easeIn});
            TweenLite.to(lilla_auditoriet, 0.6,{y: 100, alpha: 0, ease:Cubic.easeIn});
            TweenLite.to(zoomin, 0.6, {y: 280, ease:Cubic.easeIn});
            TweenLite.to(zoomout, 0.6, {y: 280, ease:Cubic.easeIn});
            audiDrop = true;
}

private function seminarieEaseDown():void
{
```

```
            TweenLite.to(pacius, 0.6, {y: 280, ease:Cubic.easeIn});
            TweenLite.to(stora_torget, 0.6, {y: 325, ease:Cubic.easeIn});
            TweenLite.to(b3320, 0.6, {y: 190, alpha: 1, ease:Cubic.easeIn});
            TweenLite.to(d4106, 0.6, {y: 235, alpha: 1, ease:Cubic.easeIn});
            TweenLite.to(zoomin, 0.6, {y: 370, ease:Cubic.easeIn});
            TweenLite.to(zoomout, 0.6, {y: 370, ease:Cubic.easeIn});
            seminarieDrop =  false;
            if (panoChooser == "B3320")
            {
                        TweenLite.to(b3320, 0.6, {y: 190, alpha: 0.6, ease:Cubic.easeIn});
            }
            else if (panoChooser == "D4106")
            {
                        TweenLite.to(d4106, 0.6, {y: 235, alpha: 0.6, ease:Cubic.easeIn});
            }

}

private function seminarieEaseUp():void
{
            TweenLite.to(pacius, 0.6, {y: 190, ease:Cubic.easeIn});
            TweenLite.to(stora_torget, 0.6, {y: 235, ease:Cubic.easeIn});
            TweenLite.to(b3320, 0.6, {y: 145, alpha: 0, ease:Cubic.easeIn});
            TweenLite.to(d4106, 0.6, {y: 145, alpha: 0, ease:Cubic.easeIn});
            TweenLite.to(zoomin, 0.6, {y: 280, ease:Cubic.easeIn});
            TweenLite.to(zoomout, 0.6, {y: 280, ease:Cubic.easeIn});
            seminarieDrop = true;
}

/*          //          Buttons //              */

// Panorama Navigations
public function stora_auditoriet_click(event:MouseEvent):void
{
            panoSphere.material.bitmap.dispose();
            panoSphere.material.destroy();
            scene.removeChild(panoSphere);
            panoChooser = "stora_auditoriet";
            generateScene();
            camera.zoom = 1.5;
            camera.rotationX = 2;
            camera.rotationY = 11;
            stora_a_info.x = (stage.width / 2) - (stora_a_info.width / 2);
            stora_a_info.y = stage.height - stora_a_info.height;
            activePanorama();
}

public function lilla_auditoriet_click(event:MouseEvent):void
{
            panoSphere.material.bitmap.dispose();
            panoSphere.material.destroy();
            scene.removeChild(panoSphere);
            panoChooser = "lilla_auditoriet";
            generateScene();
```

```
            camera.zoom = 1.5;
            camera.rotationX = 10;
            camera.rotationY = 105;
            lilla_a_info.x = (stage.width / 2) - (lilla_a_info.width / 2);
            lilla_a_info.y = stage.height - lilla_a_info.height;
            activePanorama();
}

public function d4106_click(event:MouseEvent):void
{
            panoSphere.material.bitmap.dispose();
            panoSphere.material.destroy();
            scene.removeChild(panoSphere);
            panoChooser = "D4106";
            generateScene();
            camera.zoom = 1.5;
            camera.rotationX = 6;
            camera.rotationY = 128;
            d4106_info.x = (stage.width / 2) - (d4106_info.width / 2);
            d4106_info.y = stage.height - d4106_info.height;
            activePanorama();
}

public function b3320_click(event:MouseEvent):void
{
            panoSphere.material.bitmap.dispose();
            panoSphere.material.destroy();
            scene.removeChild(panoSphere);
            panoChooser = "B3320";
            generateScene();
            camera.zoom = 1.5;
            camera.rotationX = 5;
            camera.rotationY = 204;
            b3320_info.x = (stage.width / 2) - (b3320_info.width / 2);
            b3320_info.y = stage.height - b3320_info.height;
            activePanorama();
}

public function pacius_click(event:MouseEvent):void
{
            panoSphere.material.bitmap.dispose();
            panoSphere.material.destroy();
            scene.removeChild(panoSphere);
            panoChooser = "pacius";
            generateScene();
            camera.zoom = 1.5;
            camera.rotationX = 8;
            camera.rotationY = 13;
            pacius_info.x = (stage.width / 2) - (pacius_info.width / 2);
            pacius_info.y = stage.height - pacius_info.height;
            activePanorama();
}

public function stora_torget_click(event:MouseEvent):void
```

```
{
            panoSphere.material.bitmap.dispose();
            panoSphere.material.destroy();
            scene.removeChild(panoSphere);
            panoChooser = "stora_torget";
            generateScene();
            camera.zoom = 1.5;
            camera.rotationY = -65;
            camera.rotationX = -8;
            stora_torget_info.x = (stage.width / 2) - (stora_torget_info.width / 2);
            stora_torget_info.y = stage.height - stora_torget_info.height;
            activePanorama();
}

public function onZoom():void
{
            if(camera.zoom <= 0.825)
            {
                        zoomout.enabled = false;
                        zoomout.alpha = 0.5;
                        zoomin.enabled = true;
                        zoomin.alpha = 1;
            }
            else if(camera.zoom >= 2.927)
            {
                        zoomout.enabled = true;
                        zoomout.alpha = 1;
                        zoomin.enabled = false;
                        zoomin.alpha = 0.5;
            }
            else
            {
                        zoomout.enabled = true;
                        zoomout.alpha = 1;
                        zoomin.enabled = true;
                        zoomin.alpha = 1;
            }

}

public function onMouseWheel(event:MouseEvent):void
{
            camera.zoom +=event.delta/10;

            if(camera.zoom < 0.825)
            {
                        camera.zoom = 0.825;
            }
            else if(camera.zoom > 2.927)
            {
                        camera.zoom = 2.927;
            }
}
```

```
//          Zoom functions
public function zoomOutClick(event:MouseEvent):void
{
            zoomNumber = camera.zoom - (camera.zoom / 2) / 2;
            TweenLite.to(camera, 0.25,{ zoom: zoomNumber, ease:Linear.easeNone});
            if (camera.zoom < 1.35)
            {
                        zoomout.enabled = false;
                        zoomout.alpha = 0.5;
            }
            else if (camera.zoom > 1.7)
            {
                        zoomin.enabled = true;
                        zoomin.alpha = 1;
            }
}

public function zoomInClick(event:MouseEvent):void
{
            zoomNumber = camera.zoom + (camera.zoom / 2) / 2;
            TweenLite.to(camera, 0.25,{ zoom: zoomNumber, ease:Linear.easeNone});
            if (camera.zoom > 1.7)
            {
                        zoomin.enabled = false;
                        zoomin.alpha = 0.5;
            }
            else if (camera.zoom < 1.35)
            {
                        zoomout.enabled = true;
                        zoomout.alpha = 1;
            }
}

// Popup information
protected function popup():void
{
            show_info.x = (stage.width / 2) - (show_info.width / 2); // center img
            show_info.y = (stage.height - show_info.height);
            popupWindow.x = (stage.width / 2) - (popupWindow.width / 2);
            popupWindow.y = (stage.height / 2) - (popupWindow.height / 2);
            panoChooser = "stora_torget"
            activePanorama();
            stora_torget_info.alpha = 0;
            show_info.alpha = 1;
}

private function copyright(event:ContextMenuEvent):void
{
            popupWindow.addEventListener(MouseEvent.CLICK, closePopup);
            popupWindow.x = (stage.width / 2) - (popupWindow.width / 2);
            popupWindow.y = (stage.height / 2) - (popupWindow.height / 2);
            popupWindow.visible = true;
}
```

```
private function rightClick():void
{
            myMenuItem = new ContextMenuItem("© Mikael Lindberg 2011", true)
            contextMenuCustomItems =
FlexGlobals.topLevelApplication.contextMenu.customItems;
            contextMenuCustomItems.push(myMenuItem);
}

//  -- Active Panorama Check --  //
protected function activePanorama():void
{
            if (panoChooser == "stora_auditoriet")
            {
                        stora_torget.enabled = true;
                        lilla_auditoriet.enabled = true;
                        stora_auditoriet.enabled = false;
                        d4106.enabled = true;
                        b3320.enabled = true;
                        pacius.enabled = true;

                        zoomin.alpha = 1;
                        zoomout.alpha = 1;
                        zoomin.enabled = true;
                        zoomout.enabled = true;

                        stora_torget.alpha = 1;
                        lilla_auditoriet.alpha = 1;
                        stora_auditoriet.alpha = 0.6;
                        d4106.alpha = 0;
                        b3320.alpha = 0;
                        pacius.alpha = 1;

                        show_info.alpha = 0;
                        stora_a_info.alpha = 1;
                        lilla_a_info.alpha = 0;
                        stora_torget_info.alpha = 0;
                        d4106_info.alpha = 0;
                        b3320_info.alpha = 0;
                        pacius_info.alpha = 0;

                        stora_a_info.visible = true;
                        stora_torget_info.visible = false;
                        lilla_a_info.visible = false;
                        d4106_info.visible = false;
                        b3320_info.visible = false;
                        pacius_info.visible = false;
            }
            else if (panoChooser == "lilla_auditoriet")
            {
                        stora_torget.enabled = true;
                        lilla_auditoriet.enabled = false;
                        stora_auditoriet.enabled = true;
                        d4106.enabled = true;
                        b3320.enabled = true;
```

```
            pacius.enabled = true;

            zoomin.alpha = 1;
            zoomout.alpha = 1;
            zoomin.enabled = true;
            zoomout.enabled = true;

            stora_torget.alpha = 1;
            lilla_auditoriet.alpha = 0.6;
            stora_auditoriet.alpha = 1;
            d4106.alpha = 0;
            b3320.alpha = 0;
            pacius.alpha = 1;

            show_info.alpha = 0;
            lilla_a_info.alpha = 1;
            stora_a_info.alpha = 0;
            stora_torget_info.alpha = 0;
            d4106_info.alpha = 0;
            b3320_info.alpha = 0;
            pacius_info.alpha = 0;

            stora_torget_info.visible = true;
            stora_a_info.visible = false;
            lilla_a_info.visible = false;
            d4106_info.visible = false;
            b3320_info.visible = false;
            pacius_info.visible = false;
}
else if (panoChooser == "B3320")
{
            stora_torget.enabled = true;
            lilla_auditoriet.enabled = true;
            stora_auditoriet.enabled = true;
            d4106.enabled = true;
            b3320.enabled = false;
            pacius.enabled = true;

            zoomin.alpha = 1;
            zoomout.alpha = 1;
            zoomin.enabled = true;
            zoomout.enabled = true;

            stora_torget.alpha = 1;
            lilla_auditoriet.alpha = 0;
            stora_auditoriet.alpha = 0;
            pacius.alpha = 1;
            b3320.alpha = 0.6;
            d4106.alpha = 1;

            show_info.alpha = 0;
            b3320_info.alpha = 1;
            stora_a_info.alpha = 0;
            lilla_a_info.alpha = 0;
```

```
            stora_torget_info.alpha = 0;
            d4106_info.alpha = 0;
            pacius_info.alpha = 0;

            b3320_info.visible = true;
            stora_torget_info.visible = false;
            stora_a_info.visible = false;
            lilla_a_info.visible = false;
            d4106_info.visible = false;
            pacius_info.visible = false;
}
else if (panoChooser == "D4106")
{
            stora_torget.enabled = true;
            lilla_auditoriet.enabled = true;
            stora_auditoriet.enabled = true;
            d4106.enabled = false;
            b3320.enabled = true;
            pacius.enabled = true;

            zoomin.alpha = 1;
            zoomout.alpha = 1;
            zoomin.enabled = true;
            zoomout.enabled = true;
            d4106.alpha = 0.6;
            b3320.alpha = 1;
            stora_torget.alpha = 1;
            lilla_auditoriet.alpha = 0;
            stora_auditoriet.alpha = 0;
            pacius.alpha = 1;

            show_info.alpha = 0;
            stora_a_info.alpha = 0;
            lilla_a_info.alpha = 0;
            stora_torget_info.alpha = 0;
            d4106_info.alpha = 1;
            b3320_info.alpha = 0;
            pacius_info.alpha = 0;

            d4106_info.visible = true;
            stora_torget_info.visible = false;
            stora_a_info.visible = false;
            lilla_a_info.visible = false;
            b3320_info.visible = false;
            pacius_info.visible = false;
}
else if (panoChooser == "pacius")
{
            stora_torget.enabled = true;
            lilla_auditoriet.enabled = true;
            stora_auditoriet.enabled = true;
            d4106.enabled = true;
            b3320.enabled = true;
            pacius.enabled = false;
```

```
                zoomin.alpha = 1;
                zoomout.alpha = 1;
                zoomin.enabled = true;
                zoomout.enabled = true;

                stora_torget.alpha = 1;
                pacius.alpha = 0.6;
                show_info.alpha = 0;

                pacius_info.alpha = 1;
                stora_a_info.alpha = 0;
                lilla_a_info.alpha = 0;
                stora_torget_info.alpha = 0;
                d4106_info.alpha = 0;
                b3320_info.alpha = 0;

                pacius_info.visible = true;
                stora_torget_info.visible = false;
                stora_a_info.visible = false;
                lilla_a_info.visible = false;
                d4106_info.visible = false;
                b3320_info.visible = false;

                if(!seminarieDrop)
                {
                            d4106.alpha = 1;
                            b3320.alpha = 1;
                }
                else if (!audiDrop)
                {
                            lilla_auditoriet.alpha = 1;
                            stora_auditoriet.alpha = 1;
                }
                else
                {
                            lilla_auditoriet.alpha = 0;
                            stora_auditoriet.alpha = 0;
                            d4106.alpha = 0;
                            b3320.alpha = 0;
                }
}
else if (panoChooser == "stora_torget")
{
                stora_torget.enabled = false;
                lilla_auditoriet.enabled = true;
                stora_auditoriet.enabled = true;
                d4106.enabled = true;
                b3320.enabled = true;
                pacius.enabled = true;

                zoomin.alpha = 1;
                zoomout.alpha = 1;
                zoomin.enabled = true;
```

```
                        zoomout.enabled = true;

                        stora_torget.alpha = 0.6;
                        pacius.alpha = 1;

                        show_info.alpha = 0;

                        stora_torget_info.alpha = 1;
                        stora_a_info.alpha = 0;
                        lilla_a_info.alpha = 0;
                        d4106_info.alpha = 0;
                        b3320_info.alpha = 0;
                        pacius_info.alpha = 0;

                        stora_torget_info.visible = true;
                        stora_a_info.visible = false;
                        lilla_a_info.visible = false;
                        d4106_info.visible = false;
                        b3320_info.visible = false;
                        pacius_info.visible = false;

                        stora_torget_info.x = (stage.width / 2) - (stora_torget_info.width /
2);
                        stora_torget_info.y = stage.height - stora_torget_info.height;

                        if(!seminarieDrop)
                        {
                                    d4106.alpha = 1;
                                    b3320.alpha = 1;
                        }
                        else if (!audiDrop)
                        {
                                    lilla_auditoriet.alpha = 1;
                                    stora_auditoriet.alpha = 1;
                        }
                        else
                        {
                                    lilla_auditoriet.alpha = 0;
                                    stora_auditoriet.alpha = 0;
                                    d4106.alpha = 0;
                                    b3320.alpha = 0;
                        }
            }
}

private function closeInfo(event:MouseEvent):void
{
            TweenLite.to(stora_a_info, 1, {alpha: 0, ease:Cubic.easeOut});
            TweenLite.to(lilla_a_info, 1, {alpha: 0, ease:Cubic.easeOut});
            TweenLite.to(stora_torget_info, 1, {alpha: 0, ease:Cubic.easeOut});
            TweenLite.to(d4106_info, 1, {alpha: 0, ease:Cubic.easeOut});
            TweenLite.to(b3320_info, 1, {alpha: 0, ease:Cubic.easeOut});
            TweenLite.to(pacius_info, 1, {alpha: 0, ease:Cubic.easeOut});
            TweenLite.to(show_info, 1 ,{alpha: 1, ease:Cubic.easeIn});
```

```
            var timer:Timer = new Timer(700, 1);
            timer.addEventListener(TimerEvent.TIMER, delay);
            timer.start();

            function delay(e:TimerEvent):void
            {
                        stora_a_info.visible = false;
                        lilla_a_info.visible = false;
                        stora_torget_info.visible = false;
                        d4106_info.visible = false;
                        b3320_info.visible = false;
                        pacius_info.visible = false;
            }
}

private function showInfo(event:MouseEvent):void
{
            if (panoChooser == "stora_auditoriet")
            {
                        TweenLite.to(stora_a_info, 1, {alpha: 1, ease:Cubic.easeIn});
                        TweenLite.to(show_info, 1, {alpha: 0, ease:Cubic.easeOut});
                        stora_a_info.visible = true;
            }
            else if (panoChooser == "lilla_auditoriet")
            {
                        TweenLite.to(lilla_a_info, 1, {alpha: 1, ease:Cubic.easeIn});
                        TweenLite.to(show_info, 1, {alpha: 0, ease:Cubic.easeOut});
                        lilla_a_info.visible = true;
            }
            else if (panoChooser == "stora_torget")
            {
                        TweenLite.to(stora_torget_info, 1, {alpha: 1, ease:Cubic.easeIn});
                        TweenLite.to(show_info, 1, {alpha: 0, ease:Cubic.easeOut});
                        stora_torget_info.visible = true;
            }
            else if (panoChooser == "D4106")
            {
                        TweenLite.to(d4106_info, 1, {alpha: 1, ease:Cubic.easeIn});
                        TweenLite.to(show_info, 1, {alpha: 0, ease:Cubic.easeOut});
                        d4106_info.visible = true;
            }
            else if (panoChooser == "B3320")
            {
                        TweenLite.to(b3320_info, 1, {alpha: 1, ease:Cubic.easeIn});
                        TweenLite.to(show_info, 1, {alpha: 0, ease:Cubic.easeOut});
                        b3320_info.visible = true;
            }
            else if (panoChooser == "pacius")
            {
                        TweenLite.to(pacius_info, 1, {alpha: 1, ease:Cubic.easeIn});
                        TweenLite.to(show_info, 1, {alpha: 0, ease:Cubic.easeOut});
                        pacius_info.visible = true;
            }
}
```

```
public function closePopup(event:MouseEvent):void
{
            popupWindow.removeEventListener(MouseEvent.CLICK, closePopup);
            popupWindow.visible =false;
}
```